• Nr. 2 - 2003 • 100.000 lei • ISSN 1453-7079

CHIP Special

PeCD
Peste 60 de scripturi PHP

Creați ușor un

Website dinamic

- Interactivitate pentru pagina dumneavoastră
- Secretele profesioniștilor la îndemâna oricui

PHP și MySQL pe înțelesul tuturor

Securitate

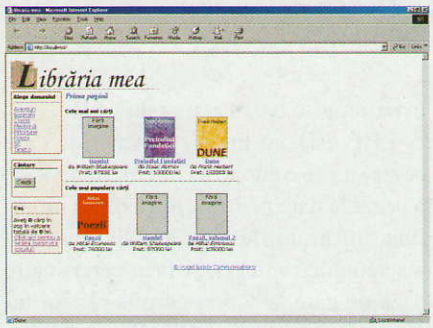
Cele mai bune
TIPS & TRICKS

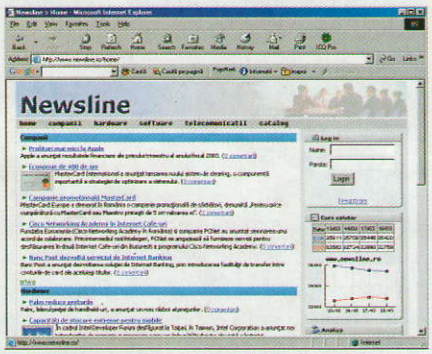
Soluția completă pentru un

magazin online

Pe CD

Apache 1.3.27 şi 2.0.44, PHP 4.3.1, MySQL 3.23 pentru Windows şi Linux; **Browsere:** Opera 7.03, Mozilla 1.2.1; **Scripturi complete:** PhpBB 2.0.4, phpMyAdmin, phpAdsNew, Gallery, PHPOpenChat, MyPhpMoney; **Editoare PHP şi HTML:** PhpEdit, Programmer Studio, Quanta, EditPlus, conText **MySQL GUI:** MySQLFront 2.5, SQLyog, JSoft Security Manager for mySQL, myFrontEnd **Acceleratoare PHP:** Turck MMCache for PHP, PHP Accelerator; **Clienți FTP:** FlashFXP, BulletProof FTP





Cuprins

Impressum

Redacţia poate fi contactată la:
Telefon: 0268-415158,418728, 0723-570511, 0744-754983;
Fax: 0268-418728; **E-mail:** redactie@chip.ro
Adresa redacţiei: 2200 - Braşov, str. N.D. Cocea nr.12
Adresa pentru corespondenţă:
2200 - Braşov, Oficiul Poştal 2, Căsuţa Poştală 4

Director General: Dan Bădescu (dan_badescu@vogelburda.ro)
Director tehnic: Daniel Dănilă-Békési (dan_danila@vogelburda.ro)
Redactor: Oana Săulescu (oana@chip.ro)
Secretar de redacţie: Cristiana Vrăbioiu (cristiana_vrabioiu@chip.ro)

Coperta: Virgil Popa (virgil@chip.ro)
Grafică, DTP: Virgil Popa (virgil@chip.ro)

Contabilitate şi administraţie: Maria Parge, Eva Szaszka, Adrian Dumitru (contabilitate@vogelburda.ro)
Reclamă: Zsolt Bodola (zsolt_bodola@vogelburda.ro), Cristian Pop (cristian_pop@vogelburda.ro)
Marketing: Leonte Mărginean (leonte_marginean@vogelburda.ro), Oana Leu (oana_leu@vogelburda.ro)
Distribuţie : Ioana Bădescu (ioana_badescu@vogelburda.ro), Ioan Soiu (iancu_soiu@vogelburda.ro)

Pregătire filme: Artúr Repro LTD.
Montaj şi tipar: Veszprémi Nyomda RT, Ungaria

Relaţii internaţionale: http://www.chip.ro/html/about/international.php3

Editura: Vogel Burda Communications SRL
Sediul editurii: 2200-Braşov Str. N.D. Cocea nr.12

ISSN 1453-7079

Soluția completă...

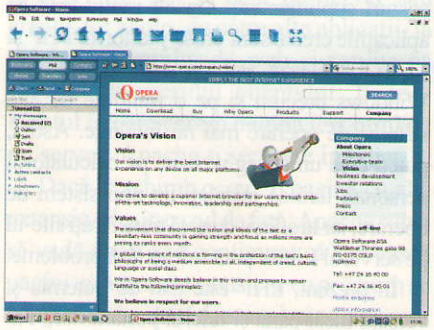
Sub acest moto am reunit pe CD o colecție a celor mai bune aplicații pentru crearea unui site web dinamic, în mare parte absolut gratuite.

Uneltele de bază

Bineînțeles, crearea unui site performant, aici înțelegând unul dinamic, precum și punerea lui în funcțiune, nu poate fi făcută fără anumite programe, dedicate. Din acest motiv, pe CD am inclus serverul web Apache, sistemul de baze de date MySQL și PHP.

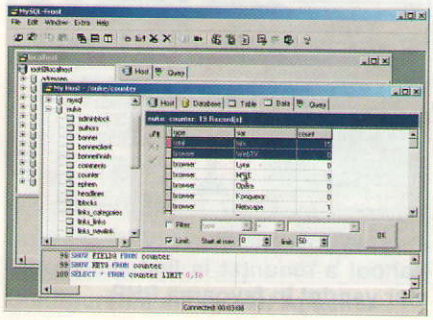
Browsere

Pentru testarea paginilor pe care le veți face este clar nevoie de un browser. Acesta a fost și unul din motivele pentru care am inclus pe CD ultimele versiuni a două browsere consacrate: Mozilla și Opera.



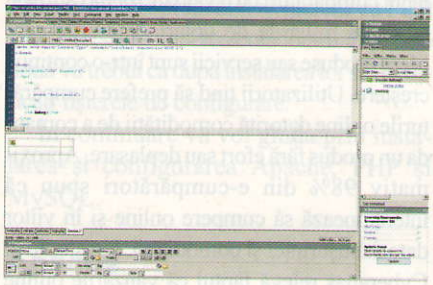
Extras

Ce ar însemna crearea/programarea unui site web fără ajutorul micilor programele care salvează timp? O muncă de luni de zile. De aceea, pe CD veți găsi numeroase astfel de utilitare: clienți vizuali pentru conectarea și lucrul cu baza de date, editoare PHP, drivere MySQL pentru conectarea din .Net, Java și Python.



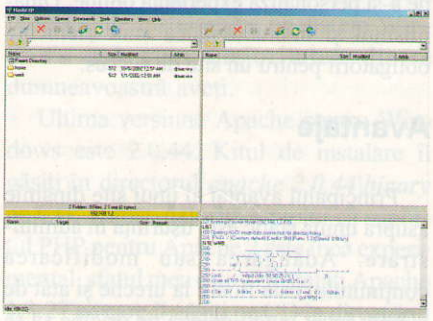
Macromedia

Nu este designer web care să nu fi auzit de firma Macromedia și produsele sale. Acestea ușurează foarte mult munca de creare, programare și finalizare a unui site web, datorită unor unelte ca Homesite, Flahsh MX, Dreamweaver, ca să amintim numai câteva din versiunile de probă incluse pe CD-ul atașat revistei. Folosind Homesite 5 sau Dreamweaver MX, crearea de site-uri web dinamice va fi mai ușoară deoarece amândouă oferă tag-tips și autocomplete pentru HTML și PHP.



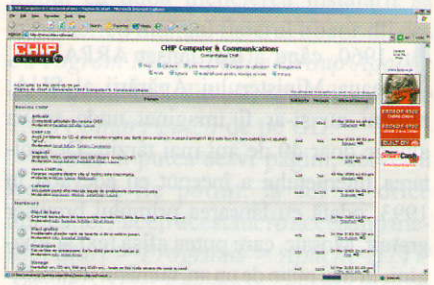
Clienți FTP

După realizarea și testarea site-ului vine și momentul în care acesta trebuie transferat pe un server conectat la Internet. Pe CD veți găsi cei mai populari clienți de FTP cu care veți putea realiza transferul rapid și ușor.



Scripturi complete

Am inclus și o serie de scripturi complete, gratuite, pe care le puteți utiliza pentru a vă îmbogăți site-ul. Printre cele mai importante se numără PhpBB 2.0.4, un forum pe care trebuie doar să îl configurați pentru a închega o comunitate pe site-ul dvs., phpMyAdmin, o unealtă cu ajutorul căreia vă puteți conecta la baza de date de oriunde ați fi, Gallery, cu care vă puteți pune pozele online, phpMyChat, un sistem de chat web-based, precum și multe altele.



Ajutor în caz de nevoie

Pe CD pe lângă Apache 1.3.27 și 2.0.44, PHP 4.3.1 și MySQL 3.23 pentru Windows și Linux veți găsi și manualele PHP și MySQL în format HTML și CHM (Windows Help).

CD-ul inclus poate fi utilizat în conformitate cu parametrii definiți în standardul Philips - Yellow Book. Editura nu-și asumă responsabilitatea asupra eventualelor pagube provocate de utilizarea CD-ului în alți parametri decât cei stabiliți în standardul menționat anterior.

NOTĂ

Interfața CD-ului CHIP Special este concepută să ruleze optim pe o placă grafică ce suportă minim o rezoluție de 800 x 600 și o adâncime a culorii de 16 biți. De aceea, nu este recomandată folosirea acesteia într-un mediu ce nu oferă minimul necesar! Interfața poate fi rulată atât sub Windows 95/98/Me, cât și sub Windows NT/2000/XP.

Din cauza multitudinii de configurații, redacția CHIP Special nu își poate asuma nici o responsabilitate în eventualitatea în care apar probleme în funcționarea interfeței și a aplicațiilor. Programele care au intrat în componența CD-ului CHIP Special au fost testate și selectate cu grijă în redacția CHIP Special. Totuși, redacția nu își poate asuma nici o responsabilitate pentru funcționarea anormală a software-ului și nici nu poate fi făcută responsabilă pentru eventualele daune produse. CD-ul CHIP Special a fost verificat împotriva virușilor cu următoarele programe antivirus (în ordine alfabetică): BitDefender Professional 6.4 (furnizat de Softwin), F-Secure AntiVirus 5.40 (furnizat de Infodesign), Kaspersky Lab Anti-Virus 4.0 (furnizat de Kaspersky Lab - Rusia) și RAV 8.5 (furnizat de GeCAD).

Pentru orice întrebări legate de aplicațiile de pe CD, vă rugăm să contactați telefonic, prin fax sau prin e-mail, autorii programelor respective.

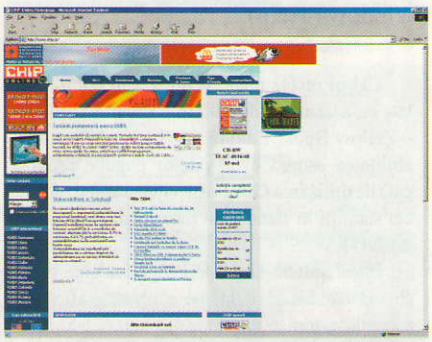
ATENȚIE! Pentru rularea corectă a interfeței CD-ului vă recomandăm setarea unei rezoluții minime de 800 x 600, o adâncime a culorii de 16 biți și folosirea opțiunii Small Fonts!

Intoducere

Încotro ne îndreptăm?

Dezvoltarea Internetului este în al treilea stadiu de dezvoltare, iar „dinamic" şi „interactiv" sunt atributele esenţiale ale oricărui site de succes.

În 1960, când a fost lansat ARPANET, reţeaua Ministerului Apărării al SUA, nimeni nu şi-ar fi imaginat unde se va ajunge doar 40 de ani mai târziu. Dezvoltarea Internetului a început cu adevărat în 1993, odată cu lansarea primului browser gratuit, Mosaic, care putea afişa text şi imagini. În mai puţin de un an, numărul utilizatorilor a ajuns la 2 milioane. David Siegel, autorul cărţii „Creating killer websites" se referă la site-urile create în această perioadă ca „Site-uri de primă generaţie", lineare, limitate ca posibilităţi dar îndeajuns de funcţionale pentru transmiterea informaţiei.



Articolele prezente pe amândouă paginile au fost introduse o singură dată şi s-au aranjat „singure": cele mai noi la început, cele mai vechi la sfârşit.

A doua generaţie a început odată cu specificaţiile pentru HTML 1 şi competiţia Netscape-Microsoft pentru dominarea pieţei. Orientarea în această perioadă a fost înspre „vizual" ca mod de comunicare.

În prezent ne aflăm în a treia generaţie, a dinamismului şi interactivităţii. Lumea progresează în pas rapid iar web designerii trebuie să ţină pasul, în această meserie evoluţia este cuvânt cheie.

Numărul internauţilor creşte de la o zi la alta, la fel ca şi aşteptările lor: ei doresc informaţie proaspătă şi posibilitatea de a interacţiona. Paginile statice nu mai sunt de ajuns pentru a asigura fidelizarea utilizatorilor. În plus, în ziua de astăzi oricine poate face un site cu ajutorul aplicaţiilor vizuale precum Macromedia DreamWeaver sau Microsoft Frontpage. Pentru a sta în faţa concurenţei, un web designer trebuie să ofere mai mult şi mai rapid decât ceilalţi.

Într-o statistică făcută pe ejobs.com, cererea de programatori HTML este în cădere liberă în timp ce limbajele server-side sunt cele mai cerute. Ca webdesigneri, ori ne adaptăm noilor cerinţei ori îngroşăm coada la Oficiul forţelor de muncă.

Se cere...

În prezent aproape jumătate de miliard din populaţia globului are acces la Internet. În 2002 39,7% din utilizatorii de Internet au făcut cumpărături cu o valoare totală estimată la 47,98 miliarde de dolari. Site-urile care oferă produse sau servicii sunt într-o continuă creştere. Utilizatorii tind să prefere cumpărăturile online datorită comodităţii de a comanda un produs fără efort sau deplasare. Aproximativ 98% din e-cumpărători spun că intenţionează să cumpere online şi în viitor datorită uşurinţei şi comodităţii. Un studiu al Cyberatlas relevă faptul că vânzările online au cel mai mare potenţial de dezvoltare din întreg sectorul economic. Site-urile firmelor nu mai sunt doar informative ci oferă utilizatorului posibilitatea de a comanda produse sau servicii. La fel şi site-urile care oferă ştiri: ele nu mai sunt demult doar simple pagini de ziar în format electronic, ci oferă utilizatorului posibilitatea de a interacţiona, de a-şi personaliza experienţa online. Dinamismul şi interactivitatea sunt elemente obligatorii pentru un site de succes.

Avantaje

Principalul avantaj al unui site dinamic asupra unuia static este uşurinţa în administrare. Adăugarea sau modificarea conţinutului este floare la ureche şi atât de simplu încât puteţi delega pe oricine să se ocupe de această sarcină. Deoarece conţinutul este stocat în baza de date sau în fişiere text, cei care se ocupă de conţinut nu au nevoie să ştie vreun pic de HTML sau organizarea sau layoutul paginilor.

De asemenea, dacă trebuie să schimbaţi logo-ul pe fiecare pagină a site-ului este de ajuns să faceţi modificarea într-un singur fişier în loc să luaţi fiecare pagină în parte, să o modificaţi şi să o uploadaţi înapoi pe server.

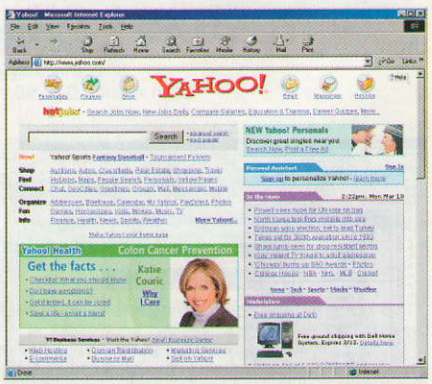
Într-un site dinamic se pot adăuga elemente de interactivitate foarte uşor. Metodele puse la dispoziţie de aplicaţiile server-side ne pot ajuta să transformăm un simplu site într-o comunitate activă.

Tendinţa este ca în viitor, toate electronicele din jurul nostru să fie conectate la Internet. Deja putem accesa site-uri wap prin intermediul telefonului mobil şi s-a inventat până şi frigiderul care face singur comenzile online. De aici se pot ridica o serie de probleme de compatibilitate. Un site creat pentru web va trebui recreat pentru a putea fi accesibil şi de pe telefonul mobil sau ultimul răcnet de mp3 player cu conectare la Internet. Având conţinutul separat de structură sau de prezentare, într-o bază de date, se poate crea foarte uşor un duplicat al site-ului original, cu un format potrivit destinaţiei.

De ce PHP şi MySQL?

Cel mai puternic motiv pentru folosirea PHP şi MySQL ca unelte pentru crearea unui site dinamic este faptul că amândouă sunt open-source şi oricine le poate utiliza fără costuri suplimentare. Un alt motiv este că aplicaţiile create sunt cross-platform: PHP şi MySQL rulează atât pe sisteme Linux şi Windows precum şi pe o mulţime de alte sisteme de operare mai rar întâlnite. Astfel, puteţi crea un întreg website pe calculatorul personal, fără să instalaţi un nou sistem de operare, iar la sfârşit doar să transferaţi site-ul pe serverul Linux şi să ruleze fără probleme.

În prezent, PHP este atât de puternic şi versatil încât până şi liderul portalurilor, Yahoo!, a renunţat la limbajul său proprietar, yscript, în favoarea PHP. Aceasta nu înseamnă că PHP este complicat sau dificil: dimpotrivă, PHP şi MySQL sunt foarte uşor de învăţat, chiar şi pentru cei nefamiliarizaţi cu programarea sau bazele de date. Acest Special doreşte să demonstreze acest lucru - tot ce va trebui să faceţi este să citiţi şi să exersaţi chiar voi toate exemplele prezentate în continuare.



Yahoo! a renunţat la limbajul proprietar yscript în favoarea PHP

Instalare şi cerinţe de sistem

Cei trei magnifici

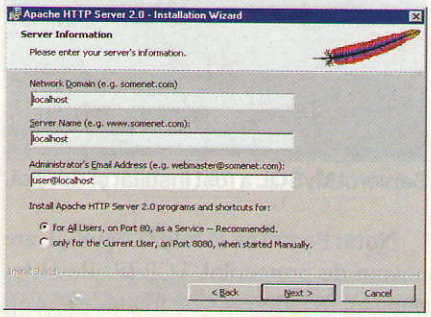
Instalarea celor trei componente necesare funcţionării unui server web poate da bătăi de cap chiar şi unui utilizator experimentat în ale calculatoarelor. Omiterea adaugării unei linii într-un fişier de configurare vă poate scoate peri albi încercând să aflaţi de ce lucrurile nu funcţioneaza aşa cum ar trebui.

Dar de ce e atât de complicat, aţi putea întreba? Ei bine, pentru vă explica acest lucru, trebuie întâi să vă povestesc ce se petrece de fapt pe server atunci când scrieţi în bara de adrese a browserului *http://www.chip.ro/1.php*, spre exemplu. Când apăsaţi Enter, browserul trimite către serverul de HTTP http://www.chip.ro o cerere: „afişează-mi pagina 1.php a site-ului www.chip.ro". Acest server este Apache. Apache ştie să servească (de aceea se şi numeşte server) browserului doar pagini HTML. Paginile HTML pot fi deja pe server sau create, tot acolo, de către alte aplicaţii, la cererea serverului de web şi servite ca HTML browserului clientului. În momentul în care cerem pagina test.html de pe server, Apache o serveşte imediat.

Dacă în schimb cerem o pagină cu altă extensie, php în cazul de faţă, Apache caută să vadă dacă este configurat să servească pagini cu această extensie şi ce program se ocupă de ele. Astfel, atunci când cerem o pagină cu extensia php, serverul Apache va trimite mai departe cererea către PHP, „se cere un fişier PHP, ocupă-te de această cerere şi dă-mi înapoi o pagină pe care să o afişez clientului". PHP preia cererea, rulează codul din program şi dacă vede că se cer informaţii din baza de date MySQL, o accesează, extrage informaţiile cerute şi construieşte o pagină HTML pe care o va trimite serverului Apache pentru ca acesta să o servească clientului.

Un exemplu concret: prima pagină a site-ului chip.ro este un fişier PHP care ia din baza de date cele mai noi ştiri adăugate de redactori şi le trimite înapoi către Apache care le afişează vizitatorului. Dacă vă uitaţi în sursa paginii nu veţi vedea decât cod HTML. PHP rulează „în umbră", pe server (de aceea se şi numeşte limbaj „server side") şi conlucrează cu Apache astfel încât acesta din urmă să poată afişa clientului un rezultat interpretabil de către browser - în acest caz o pagină HTML. Apache nu este predefinit să lucreze cu PHP şi viceversa. În vederea colaborarii între cele două va trebui ca după instalarea lor să modificati fişierele de configurare.

În continuare vă voi ghida prin instalarea şi configurarea Apache, PHP şi MySQL.



Configurarea serverului Apache.

Apache

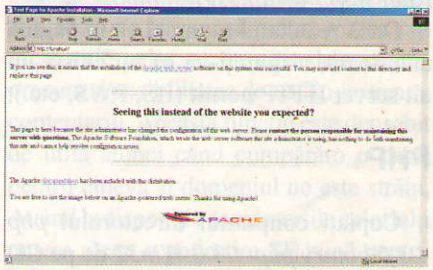
Pentru un site aveţi în primul rând nevoie de un server HTTP. Alegerea noastră s-a îndreptat spre Apache datorită flexibilităţii sale, portabilităţii, siguranţei şi extensibilităţii. Pe CD-ul alăturat puteţi găsi două versiuni Apache, pentru cele mai populare sisteme de operare. Vă voi ghida în continuare prin procesul de instalare pe un sistem Windows aşa cum majoritatea dintre dumneavoastră aveţi.

Ultima versiune Apache pentru Windows este 2.0.44. Kitul de instalare îl găsiţi în directorul *apache 2.0.44\binary* pe CD. Totuşi, datorită faptului că suportul PHP pentru Apache 2 este încă experimental, sfatul meu este să instalaţi Apache 1.3.27 pe care îl găsiţi pe CD în directorul *apache 1.3.27\binary*. Vă va fi mult mai uşor să instalaţi această versiune dar folosiţi-o doar pentru uzul personal, pentru a putea face pagini dinamice fără să fiţi nevoiţi să vă cumpăraţi hosting la un ISP sau să pierdeţi timp cu configurări exhaustive. Nu folosiţi serverul Apache pe Windows 95 sau 98 pentru a vă pune online site-urile, ci doar pentru a le crea în linişte în vederea publicării lor ulterioare pe un server securizat.

În timpul instalării Apache, în momentul în care vi se vor cere informaţii despre server, folosiţi următoarele date:
Network domain: localhost
Server name: localhost
Administrator's Email Address: adresa dvs. de mail.

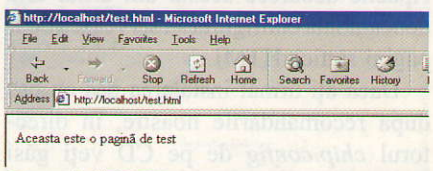
Notă: Dacă instalaţi Apache 2, va trebui ca după încheierea instalării să suprascrieţi manual fişierul libapr.dll din *c\Program Files\Apache Group\Apache 2\ bin* cu cel din directorul *apache 2.0.44\binary\patch* de pe CD.

Pentru a putea servi pagini, serverul trebuie să fie pornit, la fel ca orice aplicaţie. Faceţi acest lucru accesând meniul *Start > Programs > Apache HTTP Server > Start Apache in console* care va porni programul într-o consola. Pentru a testa, deschideţi un browser şi accesaţi adresa *http://localhost*. Dacă pagina afişată arată ca în imaginea alăturată, serverul este pornit şi funcţionează.



Prima pagină servită de server după instalare.

Pentru a nu fi nevoiţi ca de fiecare dată când vreţi să aveti serverul pornit să rulaţi programul manual, puteţi să îl porniţi ca serviciu şi el va porni automat odată cu Windows. În Windows NT, 2000 şi XP, odată cu instalarea programului, este instalat şi serviciul Apache. Îl puteţi seta să pornească automat accesând Services (Windows Service Control Manager) din Control Panel (Administrative Tools în Windows 2000). În Windows 95/98 există suport pentru servicii însă este doar experimental. Pentru ca Apache să pornească automat la fiecare repornire a calculatorului, trebuie să îl instalaţi întâi ca atare. Folosiţi o consolă MS-DOS prompt pentru a intra în directorul *C:\Program Files\Apache Group\Apache* şi a tasta apache -i -n „apache". Dacă mesajul primit este „The



O pagină HTML scrisă de utilizator şi servită de Apache.

apache service has been installed successfully." puteți să și porniți imediat serviciul (fără a reboota) scriind în consolă:

```
apache -n „apache" -k start.
```

Dacă vă hotarâți să nu mai folosiți serviciul Apache, tastați în command prompt apache -u -n „apache" și serviciul va fi dezinstalat, iar programul nu va mai fi pornit automat odata cu Windows.

Document root

Document root este directorul în care află fișierele HTML și PHP ce vor fi servite de server. În cazul de față el este directorul *C:\Program Files\Apache Group\Apache\htdocs*. Orice fișier puneți în document root va fi disponibil la adresa *http://localhost/numefisier*. Puteți testa acest lucru creând o pagină test.html în care să scrieți:

Aceasta este o pagină de test

Copiați fișierul test.html în document root și tastați în bara de adrese a browserului *http://localhost/test.html*.

Dacă rezultatul nu arată ca în pagina anterioară, asigurați-vă că nu exista un alt server HTTP pornit (IIS, PWS, etc.).

PHP

Copiați conținutul directorului *php 4.3.0\binary\php-4.3.0-Win32* de pe CD în *c:\php*, iar apoi fișierul *C:\php\php 4ts.dll* în *c:\windows\system*.

În continuare, pentru a vă ușura instalarea puteți copia fișierul *php.ini* din directorul *chip.config* de pe CD în *c:\windows*.

Alternativ, dacă doriți să faceți configurarea manual, deschideți fișierul *c:\php\php.ini-dist* cu Notepad, căutați textul doc_root= și schimbați-l cu doc_root="C:\Program Files\Apache Group\Apache\htdocs". Căutați apoi linia extension_dir = și schimbați-o cu extension_dir=„C:\php". Salvati fișierul, redenumiți-l php.ini și copiați-l în directorul Windows.

Apache+PHP

Încă nu vom putea accesa PHP din Apache deoarece acesta din urmă este deocamdată configurat să servească doar pagini statice HTML.

Dacă ați urmat instalarea pas cu pas dupa recomandările noastre, în directorul *chip.config* de pe CD veți găsi fișierul de configurare pentru Apache, *httpd.conf*, gata pregătit, fiind suficient să suprascrieți cu acesta corespondentul său din directorul *C:\Program Files\Apache Group\Apache\conf*. Dacă doriți să faceți configurarea manual (în cazul în care ați instalat PHP în alt director decât cel indicat aici), deschideți cu Notepad fisierul *C:\Program Files\Apache Group\Apache\conf\httpd.conf* și adăugați următoarele 3 linii:

```
LoadModule php4_module c:/php/sapi/
php4apache.dll
AddModule mod_php4.c
AddType application/x-httpd-php .php
```

Căutați apoi linia DirectoryIndex index.html și modificați-o în DirectoryIndex index.html index.php. Pentru ca schimbările să fie recunoscute va trebui să reporniți Apache.

Să testăm dacă totul funcționează cum trebuie. Creați cu Notepad un document text în care scrieți:

test.php

```
<?
print "Acesta este un test. Apache+PHP
funcționează";
?>
```

Salvați-l cu numele *test.php* și accesați URL-ul *http://localhost/test.php*.

Primul script PHP!

Atenție! Dacă Windows-ul dumneavoastră este setat să ascundă extensiile pentru fișierele recunoscute, documentul creat cu Notepad va fi de fapt numit *test.php.txt* și adresa http://localhost/test.php nu va putea fi accesată deoarece *test.php* nu există. În cazul acesta aveți două soluții: ori continuați să lucrați cu Notepad, dar va trebui să dezactivați ascunderea extensiilor pentru fișierele cunoscute, ori folosiți un editor PHP din cele puse la dispoziție pe CD.

MySQL

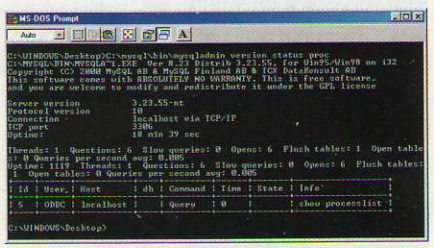
Pentru a instala ultima versiune MySQL pentru Windows nu este nevoie decât să porniți *setup.exe* din directorul *mysql 3.23\binary*. Deoarece MySQL este un server, ca și Apache, va trebui întâi să-l porniți dacă doriți să lucrați cu el. Pentru a-l porni, rulați *C:\mysql\bin\ mysqld.exe* pe Windows 95/98 sau *C:\mysql\bin\mysqld-nt.exe* pe NT/2000/XP.

Ca serverul de MySQL să pornească automat la fiecare repornire a Windows, puteți adăuga un shortcut în *Start Menu/Start up* către *C:\mysql\bin\mysqld.exe* (respectiv *mysqld-nt.exe*). Este recomandat ca pe Windows NT/2000/XP să setați serviciul MySQL să pornească automat.

Să testăm: după ce ați pornit serverul mysqld scrieți următoarea linie în command prompt:

```
C:\mysql\bin\mysqladmin version sta-
tus proc
```

Dacă output-ul este asemănător celui din fereastra alăturată, serverul MySQL funcționează.



Serverul MySQL a fost instalat și rulează.

Notă: Pentru ca să ruleze, MySQL are nevoie de protocolul TCP/IP. Pe sistemele cu Windows NT4 trebuie să aveți instalat Service Pack 3.

Dacă nu merge...

Note pentru instalarea și rularea Apache

Nu funcționează?

Dacă atunci când accesați adresa http://localhost nu vedeți o pagină cu mesajul "If you can see this, it means that the installation of the Apache web server software on this system was successful.", verificați întâi dacă serverul este pornit și dacă nu cumva aveți alt server HTTP (PWS, IIS) care rulează.

Windows 95

Dacă aveți Windows 95 trebuie să downloadați și să instalați Windows Socket 2 Update de la Microsoft înainte de a începe instalarea Apache.Windows NT 4.0, 98, ME și 2000 nu au nevoie de acest update. Adresa de download este: *http://www.microsoft.com/windows95/downloads/contents/wuadmintools/s_wunetworkingtools/w95sockets2/default.asp*

Windows XP

Pentru a putea folosi Apache pe sistemele Windows XP trebuie să aveți Windows XP Service Pack 1 instalat. Îl puteți obține de la *http://www.microsoft.com/windowsxp/pro/downloads/servicepacks/sp1/default.asp*

Atenție !

Pe unele sisteme poate fi necesar să copiați fișierele php4apache.dll și php4ts.dll din directorul /php/ în directorul /windows.

Organizarea preliminară

Cu creionul pe hârtie

Regula numărul 1 în programare este organizarea preliminară. De acesta depinde succesul acţiunii la care ne vom înhăma. Pentru a scrie cod bun trebuie să ştim destul de clar care ne sunt obiectivele înainte de a scrie chiar şi un tag HTML. Câteva minute de organizare preliminară ne pot scuti de revenirea la cod pentru a aduce modificări sau chiar rescrierea întregii aplicaţii.

Organizarea preliminară este o prac-tică obişnuită în firmele de software şi ar trebui să fie legea de căpătâi a oricărui programator web. Dacă ştim dinainte cum vrem să funcţioneze o aplicaţie, când trecem la scrisul codului com putea să ne concentrăm exclusiv asupra programării fără să avem alte preocupări. De aceea vom lua un creion şi o hârtie şi vom descrie fiecare pagină a site-ului, funcţionalităţile ei şi modul în care utilizatorul va interacţiona cu elementele acesteia.

Să definim întâi ce dorim să obţinem de la site-ul nostru. Dorim ca vizitatorii să cumpere cărţi, la fel ca într-un magazin „real". Ei vor putea să se plimbe prin spaţiul virtual, să vadă copertele cărţilor, să răsfoiască două-trei pagini, să îşi pună fiecare carte într-un coş de cumpărături şi la sfârşit să se ducă la casă pentru a plăti. Fiecare carte va fi însoţită şi de o scurtă descriere deoarece nu vom avea un librar care să facă acest lucru pentru noi. La fel ca la librărie, cărţile vor fi aranjate după domeniul căruia aparţin pentru a le putea localiza mai uşor. Şi tot pentru că nu avem un librar căruia să îi cerem direct „Daţi-mi seria Fundaţiei de Asimov" sau „Ce cărţi aveţi despre PHP?", vom pune la dispoziţia vizitatorilor noştri un motor de căutare pentru a găsi rapid ceea ce îi interesează fără a fi nevoiţi să piardă timp preţios căutând prin rafturi. Deoarece un magazin virtual aduce foarte mult cu unul real, o să vă prezint în continuare fiecare pagină a site-ului prin analogie cu o librărie reală.

Detaliile cărţii.

Vitrina

Ne aflăm la uşa librăriei. Putem vedea rafturile de cărţi marcate sugestiv „Manuale", „Medicină", „Poezii", „Aventuri", „SF", „Calculatoare", etc. În vitrină se află câteva cărţi alese pe sprânceană de către librar: cărţile foarte populare şi noile apariţii.

La fel va fi şi prima pagină a magazinului nostru virtual: vom prezenta genurile de cărţi pe care le avem disponibile, best-seller-urile, precum şi cele mai noi intrări. Pentru că spaţiul virtual ne permite, tot de la intrare vom oferi vizitatorului şi un coş de cumpărături precum şi opţiunea de căutare pentru a găsi o carte rapid, la fel cum ne ducem într-o librărie reală direct la primul angajat să îl întrebăm atunci când căutăm o anume carte.

Pe raft

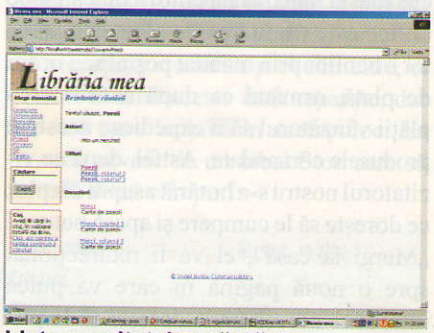
Intrăm în magazin pentru a cumpăra o carte pentru un prieten a cărui zi de naştere este duminică. Acestuia îi plac povestirile SF aşa că ne îndreptăm spre şirul de rafturi deasupra căruia scrie „SF" pentru a vedea ce am putea alege. Acest lucru este echivalent cu a da click pe unul din domeniile prezentate pe prima pagină şi a intra în pagina dedicată acelui domeniu unde se află înşirate cărţile aparţinând acestuia. Pe această nouă pagină îi vom arăta vizitatorului cărţile ordonate după autor şi titlu însoţite fiecare de o scurtă descriere şi de preţul cărţii. Titlul cărţii va fi sub formă de link astfel încât vizitatorul să poată apăsa pe el şi să vadă mai multe detalii despre cartea respectivă.

Cartea

În magazinul real putem lua o carte din raft pentru a-i vedea coperta şi a o răsfoi. Cam acelaşi lucru vom face şi în site-ul nostru. Apăsând pe link-ul reprezentat de titlul cărţii se deschide o nouă pagină în care vom prezenta detaliile cărţii, imaginea copertei, preţul şi butonul de pentru adăugarea cărţii în coş. Vom mai adăuga şi o secţiune în care să putem vedea comentariile celor ce au citit deja cartea, precum şi posibilitatea de a adăuga un nou comentariu. Aceasta funcţie este deosebit de utilă atunci când cumpărăm o carte pentru cineva şi domeniul ne este străin. Având la dispoziţie comentariile celorlalţi putem alege o povestire SF bună pentru prietenul nostru chiar dacă n-am citit-o niciodată.

Coşul de cumpărături

Atunci când un vizitator va apăsa butonul „Adaugă în coş" el va fi redirecţionat spre o nouă pagină în care îi este afişat conţinutul coşului şi două opţiuni: să continue cumpărăturile sau să meargă la casă pentru a plăti. În cazul în care doreşte să mai cumpere, aplicaţia noastră va ţine minte alegerea vizitatorului pentru a i-o putea afişa la o nouă privire în coş. Valoarea însumată a cărţilor adăugate în coş o vom afişa permanent



Lista rezultatelor căutării.

Prima pagină: cele mai noi şi mai populare cărți.

într-un colț al paginii, pentru a îmbunătăți experiența vizitatorilor pe site-ul nostru (aceştia nu vor mai trebui sa calculeze permanent în minte starea bugetului de care dispun). Utilizatorul va putea apăsa în orice moment pe coş pentru a-i revizui conținutul (cărțile alese până în momentul respectiv). Dacă totalul depăşeşte bugetul alocat el poate scoate o carte-două din coş. Dacă în schimb îşi aducem aminte în ultimul moment că un alt prieten căuta aceeaşi carte ca cea pe care doreşte să o dăruiască, utilizatorul poate modifica numărul de bucăți din coş pentru a cumpăra două în loc de una.

Căutare

Să presupunem că vizitatorul va veni pe site-ul nostru căutând o anume carte. Pentru această situație vom avea funcția de căutare. El va putea căuta după mai multe criterii: autor, numele cărții sau descriere. Rezultatul îi va fi afişat la fel ca în pagina de domeniu: titlu, autor şi descriere, cu titlul sub formă de link către pagina dedicată cărții.

Casa

Deoarece în România încă nu se pot face tranzacții online adevărate, soluția cea mai la îndemână pentru a putea vinde online este trimiterea de către cumpărător a banilor prin mandat poştal sau ordin de plată, urmând ca după confirmarea plății vânzătorul să îi expedieze acestuia produsele comandate. Astfel, după ce vizitatorul nostru s-a hotărât asupra cărților ce doreşte să le cumpere şi apasă butonul „Mergi la casă", el va fi redirecționat spre o nouă pagină în care va putea introduce adresa unde doreşte să le primească. După introducerea acestor detalii va apăsa pe „Încheie tranzacția", îi va fi afişată o pagină de mulțumire şi dumneavoastră veți fi anunțat prin email de cerere pentru a o putea onora cât mai rapid.

Administrare

Pe lângă acestea vom avea nevoie şi de o secțiune de administrare cu ajutorul căreia să adăugăm sau să scoatem cărți din rafturile magazinului virtual, să definim domenii şi să revizuim comentariile vizitatorilor (există şi posibilitatea ca vreun glumeț să adauge un comentariu nepotrivit care ar trebui şters sau modificat). Tot în secțiunea de administrare vom avea o pagină unde să vedem ultimele cereri şi să bifăm tranzacțiile efectuate cu succes după primirea banilor şi trimiterea cărților cerute către destinatar.

Structura site

După ce am terminat cu cerințele, vom trece la împărțirea pe pagini. Astfel avem:

Zona publică

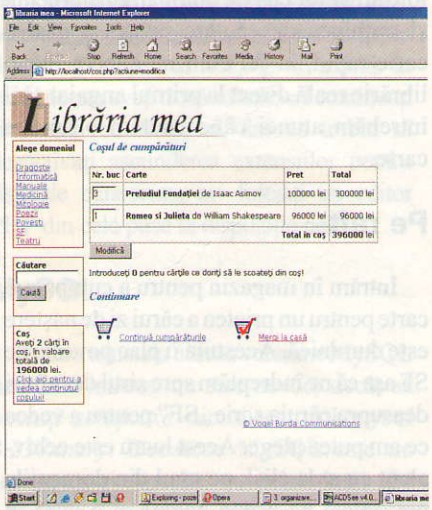
`index.php` – prima pagină

`domeniu.php` – pagina dedicată unui domeniu (SF, Aventuri, etc.) în care afişăm cărțile din domeniul respectiv

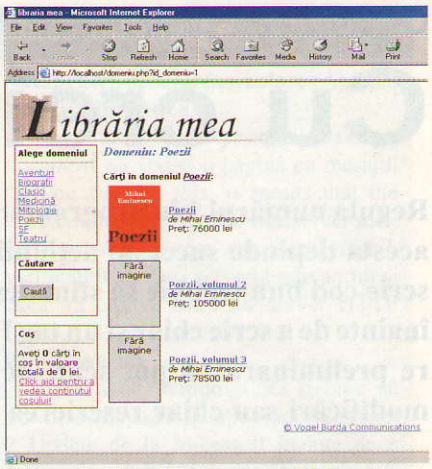
`carte.php` – pagina în care putem vedea detalii despre carte, coperta, comentariile utilizatorilor şi în care fiecare utilizator va putea adăuga un comentariu personal

`adauga_comentariu.php` – va procesa textul trimis de utilizatori şi îl va adăuga în baza de date

`cos.php` – pagina unde vizitatorul poate vedea şi edita conținutul coşului de cumpărături



Coşul de cumpărături.



Lista cărților din domeniul Poezii.

`casa.php` – formularul de introducere a datelor pentru expediere şi plată

`prelucrare.php` – pagina care va prelucra datele trimise prin formularul din casa.php: le introduce în baza de date, vă trimite un email de notificare şi apoi afişează „Multumim pentru că ați cumpărat de la noi".

Zona de administrare

Toate paginile din această zonă vor fi protejate de parolă pentru ca numai dumneavoastră să aveți acces şi se vor afla în directorul **admin**:

`index.php` – afişează un formular de login;

`login.php` – verifică informațiile din formular şi autorizează sau nu accesul mai departe, la pagina următoare;

`comenzi.php` – cu ajutorul acestei pagini vom face cele mai dese acțiuni. Ea va conține comenzile încă neonorate şi vă va da posibilitatea de a le marca. În momentul în care primiți banii pentru o comandă o puteți marca „Vândut" şi aceasta nu va mai apărea în listă. Dacă în schimb o comandă este mai veche de o lună o puteți şterge deoarece probabil cumpărătorul s-a răzgândit şi nu s-a mai dus la poştă pentru a vă trimite banii;

`adaugare.php` – pentru adăugare sau ştergere de cărți şi domenii din e-magazin;

`modificare_stergere.php` – pagina cu ajutorul căreia vom putea modifica sau şterge date de pe site: cărți, autori, domenii

`opinii.php` – afişează comentariile utilizatorilor neverificate încă de noi şi ne dă posibilitatea de a le şterge sau modifica pe cele nepotrivite.

În continuare vom trece la crearea bazei de date a magazinului virtual urmând ca apoi să facem paginile PHP din care va fi compus site-ul.

Introducere în MySQL

SQL-ul meu şi-al tău

Baza de date este coloana vertebrală a unui site dinamic. În acest capitol vom afla tot ce avem nevoie să ştim pentru a construi şi a lucra cu o bază de date.

La fel ca într-o librărie, cărţile din magazinul nostru virtual vor trebui aranjate după domeniul căruia îi aparţin, nu vom pune laolaltă într-o harababură fără sens romane de aventuri cu manuale şcolare şi cărţi de psihologie aprofundată.

În aplicaţia noastră va trebui să organizăm datele cărţilor în aşa fel încât să le putem găsi uşor după domeniu, să le ordonăm dupa titlu sau să le alegem pe cele aparţinând unui anumit autor. Soluţia cea mai flexibilă care să corespundă acestor cerinţe este baza de date.

Colecţii de date

Baza de date este, în trei cuvinte, o colecţie organizată. Ca exemple v-aş putea da cartea de telefon sau programul TV din jurnalul pe care îl citiţi dimineaţa la cafea. În cartea de telefon găsim numele si prenumele abonatului, adresa şi numărul de telefon, toate acestea ordonate după nume, pentru a căuta mai uşor o anumită persoană. Dacă am avea cartea de telefon în format electronic într-o bază de date am putea face o ordonare după numărul de telefon în ordine crescătoare pentru a afla cine are cel mai mic număr. Am putea de asemenea să aflăm câţi abonaţi au nume care încep cu Ţ, câţi dintre ei au prenumele Ion, numărul de abonaţi din oraş, pe care stradă sunt cele mai multe posturi telefonice şi aşa mai departe.

În programul TV informaţiile au următoarea structură: canalul, ora difuzarii, numele programului. Acestea sunt de obicei ordonate după canal şi ora difuzării, astfel:

Canalul A	
16:00	Desene animate: Balanel si Miaunel
18:00	Film: Tacerea mieilor
19:00	Stiri
20:00	Film: Matrix
22:00	Muzica

Canalul B	
16:00	Muzica
17:00	Documentar: Al doilea razboi mondial
18:00	Stiri
19:00	Desene animate: Mica sirena
20:00	Talkshow financiar
21:00	Film serial: StarTrek (ep. 140)
16:00	Program pentru copii

Canalul C	
17:00	Desene animate: Dexter
18:00	Film serial VIP (ep. 101)
19:00	Stiri
20:00	Film: Lord of the Rings
22:00	Documentar: Marele alb

Să ne imaginăm că tăiem cu foarfeca din ziar fiecare nume de program şi oră de difuzare.

Atunci ne-am putea ordona programele după ora difuzării astfel încât să putem distinge mai uşor ce ne interesează într-un anume interval orar:

Ora	Nume program	Canal
16:00	Desene animate:	
	Balanel si Miaunel	Canalul A
	Muzica	Canalul B
	Program pentru copii	Canalul C
17:00	Documentar: Al doilea razboi mondial	Canalul B
	Desene animate:	
	Dexter	Canalul C
18:00	Film: Tacerea mieilor	Canalul A
	Stiri	Canalul B
	Film serial VIP (ep. 101)	Canalul C
19:00	Stiri	Canalul A
	Desene animate: Lilo si Stitch	Canalul B
	Stiri	Canalul C
20:00	Film: Matrix	Canalul A
	Talkshow financiar	Canalul B
	Film: Lord of the Rings	Canalul C
21:00	Film serial: StarTrek	
	(ep. 140) Canalul B	
22:00	Muzica	Canalul A
	Documentar: Marele alb	Canalul C

Având o bază de date scăpăm şi de ziar şi de foarfecă deoarece ne putem ordona programele după plac, putem alege să vizualizăm doar programele între ora 17 şi ora 22, putem afişa pe ecranul calculatorului doar filmele din program şi aşa mai departe.

În măruntaiele bazei de date

Am spus că baza de date este o colecţie organizată, acum este momentul sa vă arat şi cum este organizată. O bază de date este compusă din unul sau mai multe tabele (*tables*) care, la rândul lor, sunt formate din înregistrări dispuse în câmpuri (*fields*).

Dacă aţi lucrat vreodată cu un program de calcul tabelar vă puteţi imagina baza de date ca fiind documentul ce conţine mai multe foi de lucru (tabelele), câmpurile ca fiind coloanele tabelelor şi o înregistrare oarecare ca fiind un rând din tabel. Să luăm exemplul cărţii de telefon. Baza de date va conţine un singur tabel cu 4 câmpuri: nume, prenume, adresă, numar de telefon.

Nume	Prenume	Adresa	Numar de telefon
Andu	Paul	Lunii 29	4567893
Baciu	Bianca	Viforului 15	3456874
Cretu	Mihai	Muncii 4	2154686
Fulg	Ramona	Republicii 14	1445328
Zinc	Ghe.	Bd. Garii 2A	6522554

Conţinutul fiecărei linii se numeşte generic înregistrare (*record*). Din prima înregistrare ştim că Andu Paul locuieşte pe Lunii 29 şi are numărul 456789. Valoarea câmpului *Nume* al primei înregistrări este Andu, iar valoarea câmpului *Prenume* a celei de-a doua înregistrări este Paul. Baza de date a departamentului contabilitate a unui magazin de papetărie va avea mai multe tabele care să conţină tipuri diferite de date. Niciunei contabile nu îi va fi util să aibă furnizorii amestecaţi cu stocurile şi cheltuielile, ci va dori să aibă aceste date separate, dar totodată puse laolaltă în acelaşi loc. Ea va avea nevoie de un tabel Stoc care să conţină două câmpuri:

Nume produs	Nr. buc. in stoc
Creioane	3511
Pixuri	5000
Plicuri A4	275

Va mai avea nevoie şi de un tabel Cheltuieli care să aibă următoarea structură:

Produs sau serviciu	Cost	Data
Decoratiuni	5.000.000 lei	20.12.2002
Reparatii	700.000 lei	03.01.2003
Jaluzele	12.000.000 lei	04 .01.2003

Pe lângă acestea îi vor mai trebui încă câteva tabele pentru salarii, vânzări, etc., pentru ca la sfârşitul anului să centralizeze datele din toate şi să facă bilanţul.

Din acest exemplu putem observa că există cazuri în care avem nevoie de mai multe tabele pentru organizarea informaţiei, tabele pe care le ţinem într-o singură bază de date pentru o accesare mai uşoară sau pentru a intermedia relaţii între ele. Dar despre asta mai târziu.

O bază de date poate avea un număr uriaş de tabele şi înregistrări. De pe siteul www.mysql.com aflăm că există la ora actuală baze de date pe MySQL Server cu 60000 de tabele şi 5 miliarde de înregistrări. Impresionant, nu? Cum ar fi fost să stocăm această cantitate de informaţie pe hârtie, în cataloage, şi să căutam prin milioane de pagini ce ne interesează?

Ce este MySQL?

MySQL este cel mai popular sistem de management pentru baze de date relaţionale deoarece este Open Source adică poate fi folosit fără să fim nevoiţi să plătim vreo sumă de bani. MySQL Server a fost creat pentru a lucra cu baze de date mai rapid decât soluţiile existente deja şi este folosit de ani buni în medii foarte solicitante.

Într-o bază de date relaţională datele sunt stocate în mai multe tabele separate, fiind astfel îmbunătăţite viteza şi flexibilitatea. Tabelele pot fi legate prin relaţii definite de noi, fiind astfel posibil să combinăm la cerere datele din mai multe tabele.

Ce înseamnă SQL?

SQL, acronimul pentru „Structured Query Language", este limbajul standard pentru comunicarea cu bazele de date. Comenzile SQL sunt folosite pentru a interacţiona cu baza de date (de exemplu să adauge, să modifice sau să şteargă datele).

Alte sisteme de baze de date care folosesc SQL sunt Microsoft SQL Server, Access, Oracle, Sybase, etc.

Baza noastră de date

Pentru a ne putea face o bază de date trebuie ca serverul MySQL să fie pornit. Dacă aţi urmat paşii descrişi în capitolul instalare, serverul MySQL ar trebui să ruleze deja, el fiind repornit automat la fiecare restartare a calculatorului.

În continuare va trebui să folosim o aplicaţie cu ajutorul căreia să comunicăm cu serverul. Aplicaţia se numeşte simplu mysql.exe şi se găseşte in acelaşi director, c:\mysql\bin\. Vom intra în command prompt apăsând pe Start, apoi pe Run şi scriind **command** în câmpul destinat programului ce urmează să fie rulat.Odata aflaţi în linie de comandă, scriem:

```
c:\mysql\bin\mysql.exe -p -u root
```

iar când ni se cere parola apăsăm ENTER.

Dacă serverul nu este pornit vom primi mesajul de eroare „Can't connect to MySQL server on 'localhost'".

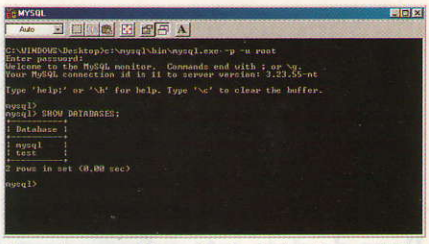
Serverul de MySQL, ca orice server care se respectă, oferă posibilitatea de a avea mai multi posesori de date pe acelaşi sistem, fiecare cu drepturile lui. Astfel, serverul unui ISP de exemplu va putea fi folosit de mai multi oameni, fiecare având mai multe baze de date şi neputând să vadă ce au ceilalţi utilizatori ai aceluiaşi server.

Utilizatorul implicit (şi cu drepturi depline, de altfel) este **root**, iar parola iniţială este goală. Vom lucra cu aceste date pentru început dar puteţi consulta manualul MySQL inclus pe CD pentru a afla cum puteţi adăuga utilizatori noi sau schimba o parolă în MySQL.

SHOW

Odată autentificaţi vom putea comunica cu serverul MySQL folosind comenzi SQL. Trimiterea unei comenzi SQL către serverul de baze de date se mai numeşte şi interogare. Să facem prima noastră interogare şi să aflăm dacă există deja vreo bază de date pe server:

```
SHOW DATABASES;
```



Acestea sunt bazele de date disponibile pe server.

Rezultatul primit de la server va arăta ca în imaginea de mai sus.

Am încheiat interogarea cu punct şi virgulă deoarece toate comenzile SQL (cu excepţia QUIT şi USE) trebuie încheiate astfel pentru a semnala serverului că am terminat de scris propoziţia şi că poate trece la procesarea cererii. Există cazuri în care o interogare poate fi foarte lungă şi atunci ar trebui să o „rupem" în câteva bucăţi mai mici, pe mai multe rânduri. MySQL ar da eroare dacă ar observa că propoziţia este neterminată aşa că nu uitaţi să încheiaţi interogările cu punct şi virgulă.

Să inspectăm una din bazele de date de pe acest server. Pentru a face acest lucru, spunem serverului cu ce bază de date dorim să interacţionăm în continuare:

```
USE test
```

Vom primi mesajul „Database changed" şi baza de date 'test' va rămâne cea în care ne vom face interogările până la deconectarea de la server sau dacă decidem să folosim o altă bază de date, cu aceeaşi comandă:

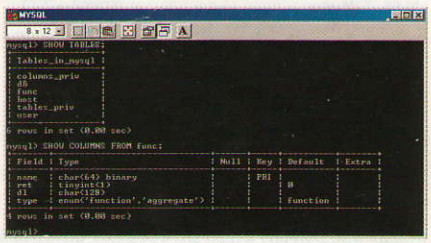
```
USE alta_baza_de_date
```

Setarea unei baze de date ca fiind cea curentă nu ne împiedică totuşi să accesăm tabele din alte baze de date în forma *nume_bază_de_date.nume_tabel*. Comanda USE este păstrată pentru compatibilitate cu Sybase.

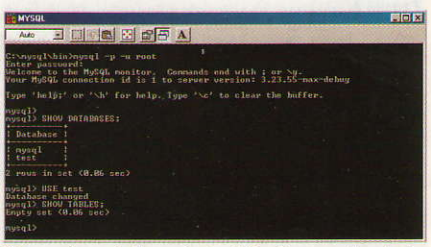
Tot comanda comanda SHOW o vom folosi pentru a vedea ce tabele există în baza de date test în care ne aflăm:

```
SHOW TABLES;
```

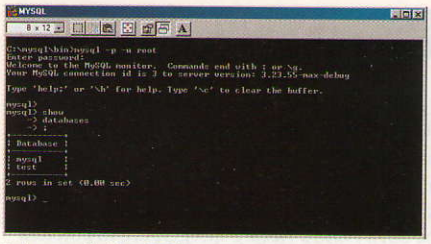
Observăm că baza de date **test** nu are nici un tabel. Să vedem dacă cealaltă are, schimbând baza de date cu care dorim să interacţionăm (USE mysql) şi repetând



Rezultatul unei interogări SHOW COLUMNS.



Nu este nici un tabel în baza de date test.



Interogările SQL se pot scrie pe mai multe rânduri cu condiția ca ultima linie să se încheie cu punct și virgulă.

comanda SHOW TABLES.

O altă utilizare a lui SHOW este SHOW COLUMNS care afișează informații despre coloanele unui tabel.

CREATE și DROP

Am spus că vom face o bază de date special pentru site-ul magazinului nostru de cărți. Să trecem atunci la treabă. Vom scrie în linia de comandă:

```
CREATE DATABASE librarie;
```

Serverul ne afișează „Query OK" deci baza de date a fost creată. Va trebui să mai spunem serverului că aceasta este baza de date cu care urmează să interacționăm în continuare:

```
USE librarie
```

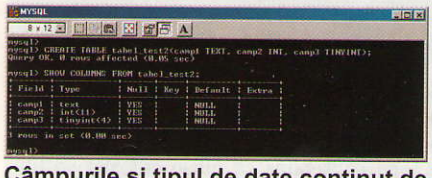
Am putea să folosim interogarea SHOW TABLES și aici, însă ea nu ne-ar arăta mai mult decât știm deja: că noua bază de date nu are încă nici un tabel. Să facem, deci, unul:

```
CREATE TABLE tabel_test
    (camp_test TEXT);
```

Temă: executați comenzile SHOW TABLES și SHOW COLUMNS pentru noua bază de date și tabelul proaspăt creat.

Să luăm cuvânt cu cuvânt propoziția: CREATE TABLE tabel_test(camp_test TEXT) înseamnă „creează tabelul tabel_test cu un câmp numit camp_text al cărui tip este TEXT. Pentru a afla ce este tipul de date și la ce folosește consultați oglinda „Tipuri de date".

Să facem încă un tabel, cu trei coloane:



Câmpurile și tipul de date conținut de acestea din tabelul tabel_test2.

```
CREATE TABLE tabel_test2(camp1
TEXT, camp2 INT, camp3 TINYINT);
```

În imaginea alăturată putem vedea mai multe informații despre fiecare câmp. Să le analizăm:

- prima coloană, Field, este numele câmpului;

- a doua, Type, este tipul de date conținut de câmp. TEXT înseamnă că datele de pe coloana respectiva vor fi texte, INT că vor fi numere întregi și TINYINT că în coloana respectivă vor fi introduse doar numere între -128 și 127;

- în a treia dacă este NULL sau nu, a patra, Key arată dacă este index, în a cincea, Default Value, este specificată valoarea implicită și ultima coloană arată ce alte proprietăți mai are câmpul.

Să revenim la magazinul nostru de cărți, să recapitulăm datele de care avem nevoie în site și să încercăm să le organizăm logic. Fiecare carte are un autor, un titlu și o scurtă descriere. Toate sunt de tip text, deci așa vor fi și tipurile de câmpuri aferente lor:

```
CREATE TABLE carti(
     autor TEXT,
     titlu TEXT,
     descriere TEXT);
```

În cazul în care nu a fost semnalată nici o eroare, tabelul nostru a fost creat. Să verificăm, de dragul exercițiului, acest lucru, cu comenzile SHOW TABLES și SHOW COLUMNS.

Pentru a șterge un tabel, o bază de date, un index sau o coloană dintr-un tabel folosim comanda DROP astfel:

```
DROP TABLE tabel_test;
DROP DATABASE librarie;
```

Deși MySQL are suport pentru diacritice și setul de caractere 8859-2, este preferabil să nu folosiți diacritice în numele bazelor de date, tabelelor sau câmpurilor. De asemenea, nu puteți folosi ca nume de tabel sau de câmp cuvintele rezervate (nume de funcții, tipuri de caractere din MySQL precum create, drop sau column).

Ar mai trebui să știți că puteți folosi nume de tabele care conțin spații dar în practică trebuie să încadrați numele între back-ticks `` (semnul back-tick îl găsiți pe tasta aflată imediat sub Escape):

```
CREATE TABLE `tabel al carui nume
are spatii`(`camp 1`, TEXT);
SHOW COLUMNS FROM `tabel al carui
nume are spatii`;
```

```
SELECT `camp 1` FROM `tabel al
carui nume are spatii`;
```

INSERT

Haideți să introducem câteva date în tabelul carti. Comanda pe care o vom folosi este INSERT și sintaxa este:

```
INSERT INTO tabel(câmp1, câmp2,
câmp3) values(valoare1, valoare2,
valoare3)
```

În românește această comandă s-ar traduce „introdu în câmpul1 valoarea1, în câmpul2 valoarea2 și în câmpul3 valoarea3", iar în format tabelar ar arăta în felul următor:

câmpul1	câmpul2	câmpul3
valoarea1	valoarea2	valoarea3

Să trecem la treabă și să exersăm comanda INSERT:

```
INSERT INTO carti(autor, titlu,
descriere) VALUES('William Shakes-
peare', 'Romeo si Julieta', 'Cea
mai frumoasă poveste de dragoste
a tuturor timpurilor');
```

Am pus ghilimele în jurul fiecărei valori ce urmează a fi introduse pentru a stabili că acela este un text unitar, un string de introdus în câmpul respectiv. Imaginați-vă cam cum ar interpreta serverul comanda noastră dacă am scrie numele autorului în forma *Shakespeare, William*. Atunci textul interogării ar fi: INSERT INTO carti(autor, titlu) VALUES(Shakespeare, William, Hamlet) și serverul MySQL ne-ar returna următoarea eroare: "Column count doesn't match value count at row 1" deoarece se așteaptă ca pentru două coloane să aibă de introdus tot două valori separate prin virgulă, nu trei. Dacă am fi specificat că vrem să introducem valori în toate cele trei coloane am fi primit alt mesaj de eroare deoarece MySQL nu ar fi recunoscut textul 'Shakespeare' ca fiind string și s-ar fi oprit din execuție.

Dacă dorim, putem să omitem una din coloane ca în exemplul următor unde nu adăugăm nimic în câmpul descriere:

```
INSERT INTO carti(autor, titlu)
VALUES('William Shakespeare',
'Hamlet');
```

Sintaxa lui INSERT poate fi simplificată dacă luăm în calcul toate câmpurile tabelului deoarece le putem omite, menționând doar valorile ce urmează a fi adăugate ca în exemplul următor:

```
INSERT INTO carti VALUES('Mihai Eminescu', 'Poezii', 'Cele mai frumoase poezii ale poetului național');
```

Observăm că nu am mai specificat coloanele în care urmează să introducem date, sintaxa fiind astfel mult simplificată. Dacă dorim să beneficiem de sintaxa prescurtată pentru INSERT dar nu avem o valoare pentru un câmp, putem să nu punem nimic între ghilimelele care delimitează valoarea respectivă, ca în exemplele următoare unde nu introducem nici un text (sau un text gol, dacă vreți să îi spuneți așa) în câmpul descriere:

```
INSERT INTO carti VALUES('Mihai Eminescu', 'Poezii, volumul 2', '');
INSERT INTO carti VALUES('Mihai Eminescu', 'Poezii, volumul 3', '');
INSERT INTO carti VALUES('Alexandru Mitru', 'Legendele Olimpului', '');
INSERT INTO carti VALUES('George Cosbuc', 'Fire de tort - Poezii', '');
INSERT INTO carti VALUES('Mihai Eminescu', 'Geniu pustiu', '');
```

Temă: Adăugați folosind INSERT încă 5 cărți în tabel specificând titlul, autorul și eventual o scurtă descriere.

SELECT

Acum avem baza de date, avem și câteva date introduse... dar cum le vedem? Vom folosi SELECT, cea mai importantă comandă SQL și îi vom vedea câteva din utilizări în exemplele următoare. Am aflat cum putem vedea ce tabele sunt într-o bază de date, hai să aflăm și ce conține un tabel. Să luăm primul tabel din făcut de noi și să afișăm datele din el:

```
SELECT * FROM tabel_test;
```

Ne va fi returnat următorul mesaj:

```
Empty set (0.17 sec)
```

Empty set înseamnă că tabelul nu conține date, e gol. Numărul din paranteze reprezintă secundele care i-au luat serverului ca să returneze un rezultat. Să încer-

Tipuri de date

Să nu amestecăm mere cu pere...

Tipurile de date care apar în coloanele MySQL ar putea părea o complicație inutilă la prima vedere. Oare nu am putea la o adică să adăugăm datele în ce formă avem nevoie? Lucrurile nu stau chiar așa. MySQL alocă spațiu pe disc în funcție de tipul de date specificat de utilizator. Dacă în tabelul salariați avem o coloană pentru numărul de zile de concediu legal o vom seta ca fiind TINYINT deoarece aceasta este valoarea numerică în care se încadrează (un salariat nu va avea mai mult de 127 de zile de concediu pe an!). Pentru fiecare înregistrare în coloana zile_concediu MySQL va aloca 1 byte de memorie, indiferent dacă un angajat are 2 zile de concediu și altul are 18. Dacă același câmp îl setăm ca INT, MySQL va aloca fiecărei înregistrări în coloană 4 bytes, indiferent de numărul de zile de concediu introduse și astfel pentru fiecare înregistrare (fiecare angajat) se vor pierde 3 bytes. Acesta este doar un exemplu dar puteți avea la un moment dat o bază de date cu 10 milioane de înregistrări și atunci pierderea a 3 bytes la fiecare înregistrare se traduce în câteva zeci de MB de spațiu.

Iată și tipurile de date în bazele de date MySQL, spațiul pe care îl ocupă precum și valorile minime și maxime pe care le pot avea.

Valori numerice:

Tip	Bytes	De la	Până la
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647
BIGINT	8	-9223372036854775808	9223372036854775807

Toate tipurile de mai sus au un atribut opțional (nestandard), UNSIGNED. Puteți folosi acest atribut în definirea tipului de date al unei coloane atunci când doriți să conțină doar valori pozitive. Un câmp de tip TINYINT va putea conține numere între -128 și 127 în timp ce alt câmp, TINYINT UNSIGNED va putea avea valori între 0 și 255. La fel, pentru SMALLINT valorile sunt de la -32768 până la 32767 în timp ce pentru SMALLINT UNSIGNED ele pot fi între 0 și 65535.

Dacă într-un câmp TINYINT care are valori între -128 și 127 veți încerca să introduceți o valoare mai mică de -128 ea va fi convertită în cea mai mică valoare admisă, -128. Dacă veți încerca să introduceți o valoare mai mare de 127 ea va fi convertită în cea mai mare valoare admisă de tipul câmpului, 127 în acest caz.

Dată/timp:

Column type	Format
DATETIME	'YYYY-MM-DD hh:mm:ss'
DATE	'YYYY-MM-DD'
TIMESTAMP	YYYYMMDDhhmmss
TIME	'hh:mm:ss'
YEAR	'YYYY'

Tipul de câmp TIMESTAMP oferă posibilitatea de a data automat operațiile de tip INSERT și UPDATE. El este compus implicit din 14 caractere pentru formatul 'YYYYMMDDhhmmss' dar putem să specificăm la crearea unui tabel că dorim să conțină mai puține caractere:

Timestamp

Column type	Display format
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(12)	YYMMDDHHMMSS
TIMESTAMP(10)	YYMMDDHHMM
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(6)	YYMMDD
TIMESTAMP(4)	YYMM
TIMESTAMP(2)	YY

Stringuri (șiruri)

Tipurile de string-uri în MySQL sunt BLOB, TEXT, CHAR, VARCHAR, ENUM și SET. În tabelul următor puteți vedea mărimea maximă admisă pentru cele mai folosite dintre ele precum și spațiul alocat pe disc pentru fiecare:

Tip	Mărime maximă	Bytes
TINYTEXT sau TINYBLOB	2^8-1	255
TEXT or BLOB	2^16-1 (64K-1)	65535
MEDIUMTEXT or MEDIUMBLOB	2^24-1 (16M-1)	16777215
LONGBLOB	2^32-1 (4G-1)	4294967295

Câmpul de tip BLOB poate conține o cantitate variabilă de informație, similar cu TEXT însă diferit printr-un singur aspect: căutarea într-un câmp BLOB este case sensitive (se face diferență între majuscule și minuscule), iar într-un câmp TEXT nu este.

Tipul VARCHAR este similar tipului TEXT cu deosebirea că într-o coloană de tip VARCHAR putem specifica numărul maxim de caractere admis.

căm cu celălalt tabel, în care ştim sigur că am introdus date şi să vedem cum arată acestea:

```
SELECT * FROM carti;
```

Acum putem vedea foarte clar structura şi conţinutul tabelului. Asteriscul * din SELECT * FROM carti este echivalentul lui „tot/toate", iar comanda s-ar traduce în limba română „arată-mi tot din tabelul test". Asteriscul îl putem înlocui cu numele unui câmp pentru a vedea doar conţinutul acelui câmp. Astfel:

```
SELECT titlu FROM carti;
```

ne va afişa doar conţinutul câmpului titlu:

```
+-------------------------+
| titlu                   |
+-------------------------+
| Hamlet                  |
| Hamlet                  |
| Poezii                  |
| Poezii, volumul 2       |
| Poezii, volumul 3       |
| Legendele Olimpului     |
| Fire de tort - Poezii   |
| Geniu pustiu            |
+-------------------------+
```

Putem selecta să ne fie afişate mai multe coloane, specificate de noi, în orice ordine (nu neapărat în ordinea în care se află în tabel), ca în interogarea următoare unde afişăm coloanele titlu şi autor:

```
SELECT titlu, autor FROM carti;
```

```
+-------------------------+-------------------------+
| titlu                   | autor                   |
+-------------------------+-------------------------+
| Hamlet                  | William Shakespeare     |
| Hamlet                  | William Shakespeare     |
| Poezii                  | Mihai Eminescu          |
| Poezii, volumul 2       | Mihai Eminescu          |
| Poezii, volumul 3       | Mihai Eminescu          |
| Legendele Olimpului     | Alexandru Mitru         |
| Fire de tort - Poezii   | George Cosbuc           |
| Geniu pustiu            | Mihai Eminescu          |
+-------------------------+-------------------------+
```

Putem afişa cărţile pentru pentru care câmpul descriere este gol astfel:

```
SELECT * FROM carti WHERE
descriere = '';
```

```
+-------------------------+-------------------------+-----------+
| autor                   | titlu                   | descriere |
+-------------------------+-------------------------+-----------+
| William Shakespeare     | Hamlet                  |           |
| M. Eminescu             | Poezii, volumul 2       |           |
| M. Eminescu             | Poezii, volumul 3       |           |
| Alexandru Mitru         | Legendele Olimpului     |           |
| George Cosbuc           | Fire de tort - Poezii   |           |
| M. Eminescu             | Geniu pustiu            |           |
+-------------------------+-------------------------+-----------+
```

Avem suficiente date pentru a afla câteva utilizări noi şi foarte interesante ale comenzii SELECT. Putem adăuga condiţii de selectie şi putem astfel alege să ne fie afişate doar titlurile cărţilor de Mihai Eminescu, astfel:

```
SELECT titlu FROM carti where
autor='Mihai Eminescu';
```

```
+-------------------------+
| titlu                   |
+-------------------------+
| Poezii                  |
| Poezii, volumul 2       |
| Poezii, volumul 3       |
| Geniu pustiu            |
+-------------------------+
```

Să afişăm toate volumele de poezii din baza de date, indiferent de autor. Dacă executăm interogarea:

```
SELECT * FROM carti WHERE
titlu='Poezii'
```

ne va returna doar cartea al cărui titlu este chiar „Poezii", neluându-le pe celelalte în considerare:

```
+-----------------+---------+-----------------------------------+
| autor           | titlu   | descriere                         |
+-----------------+---------+-----------------------------------+
| Mihai Eminescu  | Poezii  | Cele mai frumoase poezii ale poetului |
|                 |         | national                          |
+-----------------+---------+-----------------------------------+
```

Folosind condiţia titlu='Poezii' nu vor fi afişate volumele 2 şi 3 din colecţia de poezii de Mihai Eminescu şi nici cartea intitulată „Fire de tort - Poezii" de Coşbuc. Dacă vrem să obţinem toate cărţile care au cuvântul „poezii" în titlu putem face acest lucru înlocuind egalitatea cu operatorul LIKE şi folosind wildcard-uri. Semnul cu funcţie de wildcard într-o interogare MySQL este % (semnul * cu care probabil sunteţi deja obişnuiţi este definit în SQL ca însemnând tot/toate şi l-am folosit deja atunci când am utilizat SELECT * FROM carti pentru a afişa toate înregistrările din tabel). Iată cum funcţionează LIKE şi wildcardul %:

```
SELECT autor, titlu FROM carti
WHERE titlu LIKE 'poezii%';
```

Vor afişate toate înregistrările în care titlul cărţii începe cu cuvântul „poezii".

```
+-----------------+-------------------------+
| autor           | titlu                   |
+-----------------+-------------------------+
| Mihai Eminescu  | Poezii                  |
| Mihai Eminescu  | Poezii, volumul 2       |
| Mihai Eminescu  | Poezii, volumul 3       |
+-----------------+-------------------------+
```

MySQL tratează toate valorile dintr-un câmp de tip TEXT ca fiind case-insensitive astfel încât nu va trebui să ţineţi cont de literele mari sau mici atunci când vă referiţi într-o interogare la valoarea unui câmp. Comanda:

```
SELECT * FROM carti WHERE titlu
LIKE '%poezii';
```

va afişa toate înregistrările în care titlul cărţii se termină cu cuvântul „poezii":

```
+-------------------------+-------------------------+
| titlu                   | autor                   |
+-------------------------+-------------------------+
| Poezii                  | Mihai Eminescu          |
| Poezii, volumul 2       | Mihai Eminescu          |
| Poezii, volumul 3       | Mihai Eminescu          |
| Fire de tort - Poezii   | George Cosbuc           |
+-------------------------+-------------------------+
```

volumele *Poezii* de Mihai Eminescu şi *Fire de tort - Poezii* de George Coşbuc. Similar,

```
SELECT * FROM carti WHERE titlu
LIKE '%poezii%';
```

va afişa toate înregistrările în care titlul cărţii conţine cuvântul „poezii" oriunde în cadrul textului.

Putem extinde comanda SELECT pentru a adăuga un nou criteriu de selecţie.

Dacă dorim să alegem toate cărţile de poezii de Mihai Eminescu fără să ne fie afişat şi romanul scris de acesta sau cartea de poezii a lui George Coşbuc, vom trimite către baza de date o interogare cu două criterii: „returnează-mi toate înregistrările unde autorul este *Mihai Eminescu* şi titlul cărţii conţine cuvântul *poezii*".

```
SELECT * FROM carti WHERE
autor='Mihai Eminescu' AND titlu
LIKE '%poezii%';
```

```
+-----------------+-------------------+-----------------------------------+
| autor           | titlu             | descriere                         |
+-----------------+-------------------+-----------------------------------+
| Mihai Eminescu  | Poezii            | Cele mai frumoase poezii ale      |
|                 |                   | poetului national                 |
| Mihai Eminescu  | Poezii, volumul 2 |                                   |
| Mihai Eminescu  | Poezii, volumul 3 |                                   |
+-----------------+-------------------+-----------------------------------+
```

Dacă vrem să alegem toate cărțile care nu sunt scrise de Mihai Eminescu folosim operatorul de inegalitate (semnul !=, opus lui =):

```
SELECT autor, titlu FROM carti
WHERE autor != 'Mihai Eminescu';
```

```
+-----------------+-----------------+
| autor           | titlu           |
+-----------------+-----------------+
| William         | Hamlet          |
|  Shakespeare    |                 |
| Alexandru       | Legendele       |
|   Mitru         |   Olimpului     |
| George          | Fire de tort    |
|   Cosbuc        |   - Poezii      |
+-----------------+-----------------+
```

Putem găsi toate titlurile cărților pentru care câmpul descriere este gol astfel:

```
SELECT titlu FROM carti WHERE
descriere != '';
```

Analog, folosind operatorul NOT LIKE putem alege toate cărțile al căror titlu nu începe cu litera P:

```
SELECT autor, titlu FROM carti
WHERE titlu NOT LIKE 'P%';
```

```
+-----------------+-----------------+
| autor           | titlu           |
+-----------------+-----------------+
| William         | Hamlet          |
|  Shakespeare    |                 |
| Alexandru       | Legendele       |
|   Mitru         |   Olimpului     |
| George          | Fire de tort    |
|   Cosbuc        |   - Poezii      |
| Mihai           | Geniu pustiu    |
|   Eminescu      |                 |
+-----------------+-----------------+
```

Teme de rezolvat:

- afișați toate titlurile cărților de William Shakespeare;
- afișați toate titlurile care conțin litera T oriunde în textul titlului;
- afișați doar titlurile cărților ale căror autori cu numele care se termină cu litera U;
- afișați numele autorilor care nu se termină cu litera U;

Putem folosi ORDER BY pentru a ordona rezultatele unei selecții, crescător sau descrescător, în funcție de coloana aleasă pentru ordonare.

Iată, spre exemplu, cum putem afișa autorii din tabel în ordine crescătoare:

```
SELECT autor FROM carti ORDER BY
autor ASC;
```

```
+-----------------+
| autor           |
+-----------------+
| Alexandru Mitru |
| George Cosbuc   |
| Mihai Eminescu  |
| Mihai Eminescu  |
| Mihai Eminescu  |
| Mihai Eminescu  |
| William Shakespeare |
+-----------------+
```

Putem avea mai multe criterii de ordonare, ca în exemplul următor unde ordonarea se face după autor ascendent și, pentru fiecare autor, cărțile sunt ordonate după titlu descendent:

```
SELECT autor, titlu FROM carti
ORDER BY autor ASC, titlu DESC;
```

Putem afla câte înregistrări sunt pentru un criteriu de selecție cu ajutorul lui count(). Putem astfel afla câte înregistrări avem în total în tabel:

```
SELECT count(*) FROM carti;
```

```
+-----------+
| count(*)  |
+-----------+
|         8 |
+-----------+
```

sau câte înregistrări sunt în tabel al căror câmp 'autor' este Mihai Eminescu:

```
SELECT count(*) FROM carti WHERE
autor='Mihai Eminescu';
```

```
+-----------+
| count(*)  |
+-----------+
|         4 |
+-----------+
```

Cu ajutorul instrucțiunii GROUP BY putem „grupa" rezultatele astfel încât să nu vedem duplicatele și să vedem doar valorile unice.

Să vedem, folosind GROUP BY ce autori avem și să îi ordonăm crescător:

```
SELECT autor FROM carti GROUP BY
autor ORDER BY autor ASC;
```

```
+-----------------+
| autor           |
+-----------------+
| Alexandru Mitru |
| George Cosbuc   |
| Mihai Eminescu  |
| William Shakespeare |
+-----------------+
```

Pentru a limita numărul de rezultate returnate, folosim instrucțiunea LIMIT. Dacă avem 10000 de înregistrări și nu dorim să vedem decât primele 3, folosim LIMIT în felul următor:

```
SELECT * FROM carti LIMIT 0,3;
```

Dacă dorim să vedem înregistrările de la 3 la 7:

```
SELECT * FROM carti LIMIT 3,4;
```

iar pentru înregistrările de la 10 la 15:

```
SELECT * FROM carti LIMIT 10,5;
```

Observăm că primul număr din LIMIT este înregistrarea de la care se începe afișarea și al doilea număr este numărul de înregistrări ce urmează a fi returnate.

DELETE

DELETE se folosește pentru ștergerea înregistrărilor dintr-un tabel (ne reamintim că pentru ștergerea tabelelor și a bazelor de date se folosește DROP). Sintaxa lui DELETE este:

```
DELETE FROM table WHERE condiții;
```

Așa cum putem afișa doar înregistrările ce corespund condițiilor noastre, la fel putem condiționa și ștergerea lor în funcție de criteriile specificate.

```
SELECT * FROM carti WHERE
autor='Mihai Eminescu';
```

afișează toate cărțile de Mihai Eminescu, în timp ce:

```
DELETE FROM carti WHERE
autor='Mihai Eminescu';
```

va șterge din tabel toate înregistrările pentru care câmpul autor este Mihai Eminescu, lăsându-le pe celelalte neatinse.

Observăm două lucruri: asteriscul a dispărut, el fiind inutil. MySQL va șterge un rând întreg la solicitarea dumneavoastră, el nu va putea să șteargă (logic) o coloană sau mai multe aparținând unei înregistrări. În al

doilea rând, cu această excepție, sintaxa pentru ştergere seamănă ca două picături de apă cu sintaxa pentru selecție. La fel ca în SELECT putem folosi mai multe condiții şi am putea şterge toate înregistrările unde autorul este *Mihai Eminescu* şi titlul cărții conține cuvântul *poezii*".

```
DELETE FROM carti WHERE
autor='Mihai Eminescu' AND titlu
LIKE '%poezii%';
```

DELETE FROM carti este echivalentul de ştergere al lui SELECT * FROM carti, adică va şterge toate înregistrările din tabel. Nu îl vom pune în aplicare dar este bine de ştiut.

UPDATE

Atunci când vrem să modificăm conținutul unei înregistrări nu este nevoie să o ştergem şi să o adăugăm în varianta nouă, putem folosi UPDATE care are următoarea sintaxă:

```
UPDATE tabel SET coloana1='noua
      valoare a coloanei 1',
      coloana2= 'noua valoare a
      coloanei 2' WHERE condiții
```

UPDATE modifică conținutul unuia sau mai multor câmpuri în funcție de condițiile specificate. Să modificăm, de exemplu, în coloana autor toate înregistrările Mihai Eminescu şi să le înlocuim cu M. Eminescu:

```
UPDATE carti SET autor='M. Eines-
cu' WHERE autor='Mihai Eminescu';
```

Executați SELECT * FROM carti pentru a observa modificarea.

Condițiile pot fi extinse la fel ca în sintaxa SELECT. În următoarea interogare vom modifica câmpul descriere ale tuturor înregistrărilor al căror titlu conține cuvântul „poezii" şi câmpul descriere este gol:

```
UPDATE carti SET
    descriere='Carte de poezii'
    WHERE titlu like '%poezii%'
    and descriere='';
```

Astfel, toate cărțile fără descriere al căror titlu conținea cuvântul *poezii* au acum descrierea *Carte de poezii*.

ALTER TABLE

ALTER TABLE ne permite să schimbăm structura unui tabel existent. Putem adăuga sau şterge coloane şi indecşi, putem redenumi coloane sau chiar tabelul în sine şi putem schimba tipul de date conținut de o coloană. Să adăugăm o coloană, **data**, în care să stocăm data adăugării cărții în baza de date. Câmpul **data** al tabelului **carti** ne va fi util atunci când vom afişa vizitatorilor noutățile din magazin pe prima pagină. El nu corespunde datei la care a fost editată cartea ci datei în care aceasta a ajuns pe rafturile virtuale ale magazinului nostru. Cele mai noi 10 titluri din librărie le-am afla astfel foarte uşor folosind următoarea interogare:

```
SELECT titlu FROM carti ORDER BY
      data DESC LIMIT 0,10;
```

Să adăugăm noua coloană, la sfârşitul tabelului:

```
ALTER TABLE carti ADD dat TEXT;
```

Am scris gresit numele coloanei, să o redenumim din „dat" în „data":

```
ALTER TABLE carti
      CHANGE dat data TEXT;
```

Ne aducem aminte că există tipul DATE şi că o dată cum e 12-12-2003 ne-ar putea fi mai utilă dacă s-ar afla într-o coloană de tip DATE decât într-una de tip TEXT aşa că vom modifica tipul de date al noii coloane:

```
ALTER TABLE carti
      CHANGE data data DATE;
```

Ca să adăugăm o coloană altundeva decât la sfârşit, între descriere şi dată spre exemplu, folosim ALTER TABLE în felul următor:

```
ALTER TABLE carti ADD
      pret MEDIUMINT UNSIGNED
      AFTER descriere;
```

INDECŞI

Cel mai folosit tip de index este id-ul. Id-ul este un număr unic de identificare pentru un element distinct (un rând) al unui tabel. Având structura tabelului **carti** în forma actuală, pentru a alege „Romeo si Julieta" de William Shakespeare, interogarea ar arăta în felul următor:

```
SELECT * FROM carti WHERE
      autor='William Shakespeare'
      AND titlu='Romeo si Julieta';
```

Ce ne facem însă dacă implementăm interogarea în această formă în aplicație şi într-o bună zi primim spre vânzare aceeaşi carte dar cu un preț diferit? Ar trebui să rescriem interogarea pentru a ne adapta situației şi atunci căutarea unei cărți ar fi făcută cu interogarea:

```
SELECT * FROM carti WHERE
   autor='William Shakespeare'
   AND titlu='William Shakespeare'
   AND pret=100000;
```

Dacă am folosi un câmp care să conțină un număr unic de identificare pentru fiecare carte căutarea ar fi mult simplificată:

```
SELECT * FROM carti WHERE
   id_carte=15;
```

Un exemplu de id din viața reală îmi vine în minte: ați fost vreodată la un centru de închiriat casete video? Vă amintiți ordinea în care erau puse casetele în rafturi? Un patron de centru de închirieri numerotează fiecare casetă nou primită şi o pune la coadă, în ordinea numerotării, nu va căuta să vadă unde ar intra din punct de vedere alfabetic, mai ales că titluri precum „The Firm" ar putea fi puse şi sub litera F şi sub litera T, după cum consideră fiecare, şi atunci confuzia ar fi şi mai mare. În plus ca să adauge titluri noi în lista de casete disponibile nu va trebui să rescrie toată lista, ci va adăuga doar titlurile noi la coadă, în ordinea numerotării. Dumneavoastră puteți astfel să aflați rapid ce e nou doar aruncând o privire pe ultima pagină a ofertei. Atunci când cereți o casetă într-un centru de închirieri specificați vânzătoarei numărul sub care se găseşte titlul în listă, altfel ei i-ar lua câteva minute bune să caute un film anume din câteva sute. Într-o bază de date id-ul serveşte aceste funcții la fel ca în viața de zi cu zi.

Să modificăm tabelul carti astfel încât fiecare carte să aibă un id unic:

```
SET INSERT_ID=#;
ALTER TABLE carti ADD id_carte INT
   UNSIGNED NOT NULL AUTO_INCREMENT
   FIRST, ADD INDEX (id_carte);
```

SET INSERT_ID=# este folosit ca pentru fiecare carte deja adăugată să fie pus un id unic. De asemenea, pentru toate cărțile ce le veți adăuga de acum înainte va exista un id unic, incrementat automat.

Să ne folosim de index pentru a executa operații în tabel:

```
INSERT INTO carti(titlu) values('un
    text oarecare');
SELECT id_carte, titlu FROM carti;
```

Valoarea id_carte a acestei ultime înregistrări este 9. O folosim pentru a efectua modificări:

```
UPDATE carti SET titlu='un text
modificat' WHERE id_carte=9;
SELECT id_carte, titlu FROM carti
WHERE id_carte=9;
```

Rezultatul va fi:

```
+------------+-------------------+
| id_carte | titlu               |
+------------+-------------------+
|        9 | un text modificat |
+------------+-------------------+
```

Normalizarea

Normalizarea înseamnă structurarea tabelelor bazei de date în așa fel încât datele din acestea să ocupe cât mai puțin spațiu pe hard disc, fiind astfel deosebit de utilă. Chiar dacă spațiul nu poate părea o problemă, trebuie să luați în considerare posibilitatea de a fi nevoit să vă extindeți într-o zi peste numărul de megabiți oferit de contractul încheiat cu ISP-ul, din lipsă de spațiu. De ce să plătiți mai mult când puteți să normalizați? Normalizarea este o practică bună și vă recomand să o luați în calcul ori de câte ori faceți o bază de date.

Să studiem pentru un moment tabelul carti al bazei noastre de date. În momentul de față el este structurat în felul următor.

```
+-----------+-------------------------+
|Field      | Type                    |
+-----------+-------------------------+
|autor      | text                    |
|titlu      | text                    |
|descriere| text                      |
|pret       | mediumint(8) unsigned|
|data       | date                    |
+-----------+-------------------------+
```

Ce s-ar întâmpla dacă am avea toate cărțile scrise de Isaac Asimov? Numărul acestora depășește 400 și coloana **autor** ar conține de tot atâtea ori textul „Isaac Asimov". Ce putem face este să creăm un nou tabel pentru autori cu două câmpuri: **id_autor** și **nume_autor** și să înlocuim câmpul text **autor** din tabelul cărți cu un câmp numit tot **id_autor** în care pentru toate cărțile unui autor să punem id-ul din tabelul de autori. În acest mod câmpul va conține un număr din câteva cifre în loc de un text mult mai lung. Dacă scriem într-un document text numele „Isaac Asimov" vedem că acesta ocupă 12 bytes în timp ce un număr din 3 cifre (presupunând că id-ul autorului ar fi compus din 3 cifre) ocupă doar 3. Astfel înlocuind elementele repetitive dintr-un tabel cu un id facem o economie serioasă de spațiu.

Să facem, deci, un nou tabel: **autori,**

```
CREATE TABLE autori
      (id_autor SMALLINT UNSIGNED
      DEFAULT '0' NOT NULL
      AUTO_INCREMENT,
      nume_autor TEXT,
      PRIMARY KEY(id_autor),
      UNIQUE(id_autor),
      INDEX(id_autor));
```

și să redenumim câmpul autor din tabelul carti în id_autor.

```
ALTER TABLE carti CHANGE autor
      id_autor TEXT;
```

Nu îi vom modifica deocamdată tipul de date pe care îl conține. Ar trebui să revenim ulterior la tabelul autori, să căutăm id-ul corespondent fiecărui autor și să modificăm manual fiecare câmp în parte.

Vom reduce numărul de operații necesare cu ajutorul funcției UPDATE. Adăugăm întâi autorul în noul tabel, aflăm id-ul unic repartizat de către MySQL și apoi înlocuim în tabelul carti numele autorului cu acest id.

```
INSERT INTO autori values('',
      'Mihai Eminescu');
SELECT * FROM autori WHERE
      autor='Mihai Eminescu';
```

```
+------------+-------------------+
| id_autor | nume_autor          |
+------------+-------------------+
|        1 | Mihai Eminescu    |
+------------+-------------------+
```

```
UPDATE carti SET id_autor=1 WHERE
      id_autor='Mihai Eminescu';
INSERT INTO autori values('',
      'George Cosbuc');
SELECT * FROM autori;
```

```
+------------+-------------------+
| id_autor | nume_autor          |
+------------+-------------------+
|        1 | Mihai Eminescu    |
|        2 | George Cosbuc     |
+------------+-------------------+
```

```
UPDATE carti SET id_autor=2 WHERE
id_autor='George Cosbuc';
```

Temă: procedați în același fel cu fiecare nume de autor din tabelul cărți.

Acum că avem id-uri numerice în loc de nume în coloana id_autor, să modificăm tabelul **carti** și să transformăm tipul câmpului id_autor din TEXT în SMALLINT:

```
ALTER TABLE carti CHANGE id_autor
id_autor SMALLINT NOT NULL;
```

Cele două tabele arată acum așa:

autori:

```
+------------+-------------------+
| id_autor| nume_autor           |
+------------+-------------------+
| 1        | Mihai Eminescu     |
| 2        | George Cosbuc      |
| 3        | William Shakespeare|
| 4        | Alexandru Mitru    |
+------------+-------------------+
```

carti:

```
+------------+-------------------------+
|id_autor| titlu                       |
+------------+-------------------------+
|        3|Hamlet                     |
|        1|Poezii                     |
|        1|Poezii, volumul 2          |
|        1|Poezii, volumul 3          |
|        4|Legendele Olimpului        |
|        2|Fire de tort - Poezii|
|        1|Geniu pustiu               |
|        3|Romeo si Julieta           |
+------------+-------------------------+
```

Putem „uni" datele din cele două tabele astfel încât să ne fie afișate numele și titlul pentru fiecare înregistrare:

```
SELECT nume_autor, titlu FROM
   autori, carti WHERE
   autori.id_autor=carti.id_autor;
```

```
+-------------------+-------------------------+
| nume_autor        | titlu                   |
+-------------------+-------------------------+
| William Shakespeare | Hamlet                |
| Mihai Eminescu    | Poezii                  |
| Mihai Eminescu    | Poezii, volumul 2       |
| Mihai Eminescu    | Poezii, volumul 3       |
| Alexandru Mitru   | Legendele Olimpului     |
| George Cosbuc     | Fire de tort - Poezii |
| Mihai Eminescu    | Geniu pustiu            |
+-------------------+-------------------------+
```

Această interogare ia coloana nume_autor din tabelul autori și coloana titlu din tabelul cărți și le asociază după id-ul autorului care este unic. Dacă una din coloanele pe care vrem să le afișăm are aceeași nume în cele două tabele, o putem apela ca nume_tabel.nume_coloană astfel încât să nu existe ambiguități:

```
SELECT autori.id_autor,
   autori.nume_autor, carti.titlu
   FROM autori, carti WHERE
   autori.id_autor=carti.id_autor;
```

În aplicația noastră va trebui să avem cărțile ordonate după domenii. Nu vom

```
+-----------+-------------------------+-------------------------+
| id_autor  | nume_autor              | titlu                   |
+-----------+-------------------------+-------------------------+
| 3         | William Shakespeare     | Hamlet                  |
| 1         | Mihai Eminescu          | Poezii                  |
| 1         | Mihai Eminescu          | Poezii, volumul 2       |
| 1         | Mihai Eminescu          | Poezii, volumul 3       |
| 4         | Alexandru Mitru         | Legendele Olimpului     |
| 2         | George Cosbuc           | Fire de tort - Poezii   |
| 1         | Mihai Eminescu          | Geniu pustiu            |
+-----------+-------------------------+-------------------------+
```

amesteca romanele de război cu manualele şcolare şi cu poeziile! Ne va trebui în consecinţă o nouă coloană în tabel, pentru domeniul cărţii: Legendele Olimpului vor aparţine domeniului „Poveşti", cărţile de poezii domeniului Poezii şi aşa mai departe.

Temă: modificaţi tabelul carti pentru a adăuga o coloană **id_domeniu** şi creaţi un nou tabel **domenii** cu două câmpuri: **id_domeniu** şi **nume_domeniu**. Adăugaţi numele de domenii pentru cărţile din baza de date şi id-urile acestora în tabelul cărţi, conform cu id-urile din tabelul **domenii**.

Temă: afişaţi titlurile cărţilor din baza de date şi domeniile de care acestea aparţin (ca în exemplul de mai sus, cu tabelul autori):

```
+-------------+-------------------------+
|nume_domeniu|titlu                     |
+-------------+-------------------------+
|Teatru       |Hamlet                   |
|Poezii       |Poezii                   |
|Poezii       |Poezii, volumul 2        |
|Poezii       |Poezii, volumul 3        |
|Povesti      |Legendele Olimpului      |
|Poezii       |Fire de tort - Poezii    |
+-------------+-------------------------+
```

Deocamdată avem în baza de date tabelele **carti**, **autori** şi **domenii**. În continuare va trebui să mai adăugăm unul. Ne reamintim că la fiecare carte vizitatorii noştri au posibilitatea să-şi poată împărtăşi impresiile. Pentru aceasta vom crea un nou tabel, comentarii, ale cărui câmpuri nu le voi mai descrie deoarece numele şi tipul lor sunt de ajuns de clare:

```
CREATE TABLE comentarii (
  id_comentariu INT UNSIGNED
  DEFAULT '0' NOT NULL
  AUTO_INCREMENT,
  id_carte INT
  UNSIGNED DEFAULT '0',
  nume_utilizator TEXT,
  adresa_email TEXT,
  comentariu TEXT,
  PRIMARY KEY
  (id_comentariu),
  UNIQUE (id_comentariu),
  INDEX (id_comentariu));
```

Câmpul **id_carte** din acest tabel ne va folosi pentru a afla toate comentariile aferente unei cărţi, cu ajutorul interogării

```
SELECT * FROM comentarii WHERE
id_carte=1;.
```

Pentru urmărirea vânzărilor vom folosi două tabele, tranzactii şi vanzari. Să presupunem că Pop Ion cumpără două cărţi în 1 martie şi revine pentru a cumpăra încă alte trei cărţi în 2 martie. În data de 3 martie când vin banii pentru prima comandă noi vom şti că trebuie să trimitem nişte cărţi lui Pop Ion dar nu ştim exact care! Iată de ce, pentru fiecare comandă, indiferent de numărul cărţilor cerute, va trebui să folosim un număr unic de identificare, număr pe care îl ţinem în tabelul tranzactii alături de celelalte informaţii care ne interesează:

```
CREATE TABLE tranzactii (
  id_tranzactie INT unsigned
  NOT NULL auto_increment,
  data_tranzactie TIMESTAMP(10)
  NOT NULL,
  nume_cumparator TEXT NOT NULL,
  adresa_cumparator TEXT NOT NULL,
  UNIQUE KEY id_tranzactie
  (id_tranzactie));
```

Am setat câmpul data_tranzactie ca TIMESTAMP(10) pentru că astfel el va putea fi setat automat de MySQL la fiecare nouă înregistrare şi va conţine data inserării înregistrării în format YYYYMMDD. Executaţi următoarele două interogări:

```
INSERT INTO tranzactii
  (nume_cumparator,
  adresa_cumparator) VALUES ('Pop
  Ion', 'str. Lunii 12, Brasov');
SELECT * FROM tranzactii;
```

Tabelul vânzări va conţine informaţiile despre cărţile vândute în fiecare tranzacţie:

```
CREATE TABLE vanzari (
  id_tranzactie int unsigned NOT
  NULL default '0',
  id_carte int unsigned NOT NULL
  default '0',
  nr_buc tinyint NOT NULL default
  '0');
```

Câmpul id_tranzactie corespunde celui din tabelul tranzactii iar câmpul id_carte celui din tabelul carti. Câmpul nr_buc ne va servi atunci când un utilizator cumpără mai multe bucăţi din aceeaşi carte într-o tranzacţie şi de asemenea pentru a putea face statisticile pentru cele mai vândute cărţi. Introduceţi câteva înregistrări în tabel folosind id_tranzactie din tabelul tranzactii şi mai multe id_carte din tabelul carti:

```
INSERT INTO vanzari VALUES(1,2,1);
INSERT INTO vanzari VALUES(1,3,1);
INSERT INTO vanzari VALUES(1,4,30);
```

În tranzacţia cu id_tranzacţie 1 s-au vândut: 1 carte cu id_carte 2, 1 carte cu id_carte 3 şi 30 de cărţi cu id_carte 4.

Am învăţat în acest capitol cum să facem o bază de MySQL date şi cum să lucrăm cu ea pentru a obţine datele de care avem nevoie. Înainte de a trece la capitolul următor însă trebuie să vă spun că pe CD veţi găsi câţiva clienţi vizuali pentru serverul MySQL cu care vă veţi putea construi structura bazei de date mult mai uşor. V-am „chinuit" cu sintaxa în linia de comandă deoarece mai tarziu, când vom lucra cu baza de date prin intermediul PHP, va fi necesar să ştim cum se introduc sau se extrag date folosind comenzi SQL.

În continuare vom învăţa bazele programării PHP urmând ca apoi să folosim cunoştinţele acumulate în aceste două capitole pentru a obţine informaţii din baza de date prin intermediul PHP şi a le afişa utilizatorilor într-o pagină de web. ■

Mărimea contează?

Cât de mare poate fi un tabel?

MySQL stochează fizic datele unui tabel într-un fişier pe hard disc şi cu cât tabelul este mai mare, cu atât mărimea acestui fişier creşte. Versiunea 3.22 a MySQL are o limită de 4GB pentru mărimea unui tabel. În versiunile superioare această limită este extinsă până la 8 milioane TB pentru tipul de tabel MyISAM. Cu toate acestea, sistemele de operare pot avea propriile limitări ale mărimii fişierelor. Mărimea implicită a tabelelor MySQL este de aproximativ 4GB. Puteţi verifica mărimea maximă pentru un tabel cu ajutorul comenzilor SHOW TABLE STATUS sau myisamchk -dv table_name.

Pe platforma Windows va trebui să folosiţi sistemul de fişiere NTFS dacă doriţi să aveţi tabele mai mari de 4GB.

PHP

Programare pentru toți

Ca limbaj de programare, PHP este foarte ușor de învățat dacă elementele programării sunt explicate pe înțelesul tuturor. Acesta este unul din scopurile articolului de față: de a iniția orice novice în tainele programării PHP. Datorită similarității între limbajele de programare de uz comun, cunoscând PHP va fi floare la ureche să treceți apoi la VBScript, Java sau chiar C. Nu i-am uitat pe programatorii experimentați în alte limbaje care vor afla în acest capitol particularitățile PHP.

PHP este limbajul ideal pentru construirea de pagini web dinamice. Este ușor de învățat, open-source, poate fi rulat pe mai multe platforme și se poate conecta la mai multe tipuri de baze de date.

Cel mai important aspect al limbajului este însă posibilitatea de a fi îmbricat cu cod HTML. Putem astfel crea pagini HTML statice și din loc în loc, acolo unde este nevoie, să introducem dinamism cu ajutorul PHP. Să luăm prima pagină a site-ului www.chip.ro de exemplu. Mare parte din structura sa este compusă din cod HTML static (meniurile, tabelele, aranjamentul în pagină). Din loc în loc codul HTML este intercalat cu cod PHP care extrage din baza de date cele mai noi știri. În momentul în care apelați pagina, acest cod este parsat (analizat linie cu linie și executat) pe server și este afișată o pagină HTML fără să știți că pentru crearea ei s-a făcut o conexiune la baza de date, s-au extras informațiile de acolo și au fost ordonate pentru afișare.

Pe locuri...

Limbajul PHP s-a „născut" în 1994 din nevoia lui Rasmus Lerdorf de a afla câte persoane îi vizualizează CV-ul online. El a denumit setul de scripturi create PHP, acronimul pentru Personal Home Page. Pe parcursul următorilor trei ani limbajul a evoluat dar adevăratul succes a început să îl cunoască de când Zeev Suraski și Andi Gutmans au rescris motorul PHP de la cap la coadă, motor care poartă din versiunea 4 a PHP numele Zend, o combinație de litere din prenumele creatorilor săi: Zeev și Andi.

Fiind open-source, PHP beneficiază de suport activ din partea comunității online, acesta fiind și motivul creșterii explozive a numărului site-urilor bazate pe PHP. Între 2000 și 2002 numărul lor a crescut cu peste 13130% în timp ce numărul site-urilor bazate pe tehnologia ASP doar cu 278%, Java Server Pages cu 1594% și ColdFusion cu 429%.

Pe lângă manipularea conținutului paginilor de web, PHP poate trimite headere HTTP pentru autentificare, seta cookie-uri sau redirecționa utilizatorii. Mai mult, cu ajutorul bibliotecilor externe de funcții poate parsa fișiere XML, crea și manipula imagini, animații Shockwave Flash, PDF-uri sau se poate conecta la un server de mail iar acestea sunt doar câteva din funcțiile pe care le poate îndeplini.

Fiți gata...

În continuare vă voi îndruma prin elementele de bază ale PHP. Așezați-vă confortabil, deschideți un editor de text, asigurați-vă că serverul web este pornit și puneți-vă centurile de siguranță. Vă recomand să rulați chiar voi exemplele prezentate în continuare deoarece așa veți înțelege foarte ușor cum funcționează PHP la server și care este rezultatul în browser.

Puteți folosi orice editor de text. Notepad este bun pentru început dar puteți alege să folosiți unul din editoarele PHP de pe CD. Salvați fișierele cu extensia php, în document root (c:\Program Files\Apache Group\Apache\htdocs\) și le veți accesa în browser la adresa *http://localhost/numefisier.php*.

Start!

Programarea de orice fel, nu doar PHP, are două elemente de bază: datele și instrucțiunile. Pentru a lucra cu datele trebuie să înțelegem ce sunt variabilele și tipurile iar pentru a lucra cu instrucțiuni trebuie să aflăm ce sunt structurile de control și funcțiile.

Variabile

O variabilă este o zonă de memorie căruia i se dă un nume pentru a putea fi recunoscută ulterior și pentru a ne putea referi mai târziu la ea. Să ne imaginăm ca Ionel și Gigel au următoarea conversație:

Ionel: Poți aduna două numere?

Gigel: Da. Care este primul număr?

Ionel: Primul număr este 3. Al doilea număr este 2.

Gigel: Rezultatul adunării este 5.

Toate cele trei numere din conversația de mai sus le putem considera variabile. Gigel ține minte valoarea primului număr, 2, apoi ține minte valoarea celui de-al doilea număr, 3, după care le adună și obține rezultatul: 5. Discuția ar putea continua în felul următor:

Ionel: Cel de-al doilea număr este acum 6. Care este rezultatul adunării?

Gigel: Este 9.

Ionel a schimbat valoarea celui de-al doilea număr, rezultatul fiind de această dată 9. Putem considera numerele respective ca fiind variabile și atunci codul PHP pentru prima adunare ar fi:

```
$primul_numar = 3;
$al_doilea_numar = 2;
$total = $primul_numar +
        $al_doilea_numar;
```

Valoarea variabilei $total este suma celor două numere, 5 în acest caz. La a doua operație de adunare Gigel ține minte deja valoarea primului număr, schimbă valoarea celui de-al doilea și obține un nou rezultat:

```
$al_doilea_numar = 6;
$total = $primul_numar +
        $al_doilea_numar;
```

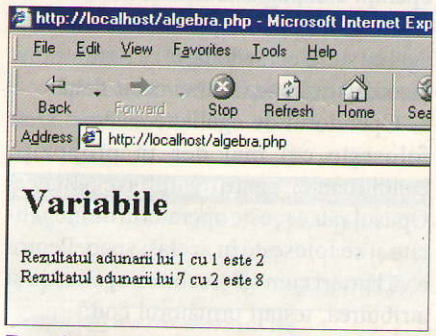
Valoarea variabilei $total se schimbă, deoarece am schimbat unul din elementele ecuației. Înțelegem acum de ce sunt numite variabilele așa: pentru că valoarea lor poate fi schimbată ori direct, cum am schimbat în a doua operație valoarea lui $al_doilea_numar din 3 în 6, ori ca

urmare a dependenței lor de alte variabile, așa cum se schimbă valoarea variabilei $total în funcție de valorile lui $primul_numar și $al_doilea_numar.

Pentru a înțelege mai bine conceptul de variabilă, putem face o comparație cu algebra. Dacă luăm un număr x și îi dăm valoarea 1, x+1=1, x+2=2. Dacă x=7, x+1=8 și x+2=9. Analog, putem scrie următorul cod PHP într-un fișier numit algebra.php:

```
<html>
<body>
<h1>Variabile</h1>

<?
// să setăm valoarea lui x ca fiind 1
$x = 1;
$rezultat = $x+1;
print "Rezultatul adunării lui $x cu 1
  este $rezultat<br>";

/* și acum să setăm valoarea lui $x ca
fiind 7 și îl vom aduna tot cu 1 */
$x = 7;
$rezultat = $x+1;
print "Rezultatul adunării lui $x cu 2
  este $rezultat<br>";
?>
</body>
</html>
```

Salvați fișierul în document root (directorul c:\Program files\Apache Group\Apache\htdocs\) și accesați *http://localhost/algebra.php*. Iată cum va arăta pagina afișată:



Rezultatul rulării în browser a fișierului algebra.php

Să disecăm codul pentru a învăța câteva lucruri:

– toate instrucțiunile PHP se termină cu punct și virgulă. Omiterea semnului „punct și virgulă" este cea mai frecventă greșeală pe care o fac programatorii începători.

– codul PHP începe întotdeauna cu <? și se termină cu ?>. El poate fi imbricat cu cod HTML după cum puteți vedea în exemplul alăturat. Puteți chiar crea pagini HTML fără pic de cod PHP în ele și să le dați extensia php. Atâta vreme cât parser-ul PHP nu „vede" tagurile <?...?>, el va trimite pagina HTML neschimbată către server.

– putem pune diacritice în cadrul unui string însă pentru ca browserul să le afișeze corect va trebui să specificăm în <head> setul de caractere folosit, la fel ca într-un document HTML.

– valoarea unei variabile poate fi schimbată după necesități (așa cum am schimbat valoarea lui $x) sau ea va fi schimbată automat în funcție de celelalte variabile de care depinde (așa cum valoarea lui $rezultat s-a schimbat în funcție de valoarea lui $x).

– pentru a afișa rezultatul folosim print, altfel valorile variabilelor ar fi fost schimbate dar nu ar fi fost afișate pe ecran. Pentru mai multe detalii privind folosirea lui print vă recomand să consultați oglinda alăturată.

– linia de text care începe cu // nu este afișată, la fel ca textul demarcat de /* ... */ și nu apar nici măcar dacă dăm View Source în browser pentru fișierul algebra.php accesat. Acestea sunt comentarii care nu sunt procesate de către server ca fiind cod executabil și nici nu sunt trimise mai departe către browser. În aplicațiile mai mari de câteva linii este util să comentăm codul pentru a ne orienta mai bine sau a explica acțiunile intreprinse. Diferența între cele două notații este că // este folosit pentru a comenta o singură linie de text în timp ce /*...*/ poate fi folosit pentru a delimita un comentariu ce se extinde pe mai multe linii. Marcatorii de comentariu se pot folosi și atunci când dorim ca o bucată de cod să nu ruleze. Puteți să testați acest lucru comentând una din liniile care conțin instrucțiunea print și rulând din nou pagina.

Numele variabilelor trebuie să conțină doar litere (a-z, A-Z și caracterele ASCII de la 127 la 255), cifre și liniuțe de subliniere (underscores) și pot începe doar cu litere sau liniuțe de subliniere. Astfel, $o_variabila, $_altaVariabila și $_inca_o_variabila_ sunt

Folosirea funcției print

Afișarea datelor

Cu ajutorul lui print putem afișa un string, o variabilă, un string ce conține variabile sau rezultatul unei funcții. Textul ce urmează a fi afișat trebuie inclus între ghilimele simple sau duble, rezultatul fiind ușor diferit. Folosind ghilimelele duble orice variabilă din cadrul stringului este parsată. Astfel, în exemplul alăturat, print "Rezultatul inmultirii lui $x cu 1 este $rezultat
"; afișează în browser Rezultatul inmultirii lui 1 cu 1 este 2. Dacă am fi folosit ghilimele simple (print '...'), variabilele $x și $rezultat nu ar fi fost parsate și output-ul ar fi fost Rezultatul inmultirii lui $x cu 1 este $rezultat. Testați acest lucru!

Putem afișa variabilele și dacă folosim ghilimele simple, "rupând" stringul și intercalându-l cu variabile, în forma următoare, folosind operatorul "." de concatenare a stringurilor (șirurilor):

```
print 'Rezultatul inmultirii lui '.$x.'
cu 1 este '.$rezultat.'<br>"
```

Pentru a afișa caracterele speciale folosite de PHP (ghilimelele duble ", ghilimelele simple ',backslash-ul \, semnul $) trebuie să le precedeți cu semnul \, astfel:

print "Semnul dolar \$, back-slash \\";

Ghilimelele trebuie precedate de semnul \ doar dacă sunt de același tip cu cele care încadrează string-ul. În plus, într-un string puteți folosi celelalte ghilimele normal.

print "Ghilimelele duble \" dintr-un string încadrat tot de ghilimele duble trebuie precedate cu semnul \\. Ghilimelele simple 'ca acestea' nu au nevoie să fie precedate deoarece se află într-un string încadrat de ghilimele duble";

Alternativ:

print 'Ghilimelele simple trebuie \'precedate\' într-un string încadrat de ghilimele simple în timp ce "ghilimelele duble" nu';

Dacă doriți să afișați doar valoarea unei variabile, puteți să nu o încadrați între ghilimele.

```
<?
$o_variabila = 1;
print $o_variabila;
$alta_variabila = "Un text oarecare";
print $alta_variabila;
?>
```

Observați că dacă rulăm acest cod, rezultatul afișat în browser va fi „ Un text oarecare", deși instrucțiunile de afișare se află pe două linii diferite. Aceasta se întâmplă deoarece, cum ziceam și în introducere, rezultatul este trimis către browser ca HTML. Dacă vrem să ne fie afișate una sub alta va trebui să intercalăm cod HTML între cele două instrucțiuni. Putem face acest lucru în două moduri: ori prin întreruperea codului PHP, așa:

```
<?
$o_variabila = 1;
print $o_variabila;
?>
<br>
<?
$alta_variabila = "Un text oarecare";
print $alta_variabila;
?>
ori prin scrierea lui &lt;br&gt; direct
în cod, așa:
<?
$o_variabila = 1;
print $o_variabila;
print "<br>";
$alta_variabila = "Un text oarecare";
print $alta_variabila;
?>
```

nume corecte în timp ce $o-noua-variabila, $2var şi $a_şaptea nu sunt. Înainte de a trece mai departe, vă dau un sfat: daţi variabilelor nume care să definească pe cât posibil destinaţia variabilei, mai ales dacă aceasta va fi folosită de mai multe ori în cadrul programului. Dacă vi se va cere să modificaţi aplicaţia după două luni de la crearea ei o să preferaţi să înghiţiţi tastatura mai degrabă decât să vă chinuiţi să aflaţi din 500 de linii de cod ce reprezintă $a1 sau $xGvHKx.

Tipuri de variabile

Variabilele pot fi de mai multe tipuri, nu doar numere cum am văzut până acum. Dacă programarea PHP ar fi fost despre algebră şi numere, n-aş fi fost aici povestindu-vă cât de minunată este! PHP are opt tipuri de variabile dintre care vi le voi descrie pe cele mai importante 4.

integer

Variabilele de tip integer sunt numere întregi: 3, 783, -56, 0, -1 sunt valori integer.

string

Un string este o succesiune de caractere (şir). Atunci când sunt folosite în codul PHP stringurile trebuie încadrate între ghilimele şi toate caracterele speciale din ele precedate cu semnul \ (detalii şi explicaţii găsiţi în oglinda **funcţia print**). Iată şi câteva exemple de stringuri:

```
$variabila = "un text oarecare";
$variabila = "10 texte oarecare";
$variabila = "10";
```

boolean

Tipul boolean defineşte o valoare de adevăr, TRUE (adevărat) sau FALSE (fals). Spre exemplu, vom scrie mai târziu o funcţie pentru secţiunea de administrare a site-ului cu ajutorul căruia vom verifica dacă utilizatorul este logat ca administrator. După ce va face toate verificările necesare, funcţia noastră va returna o valoare de adevăr: TRUE dacă este logat sau FALSE dacă nu este şi astfel vom şti dacă să îi acordăm sau nu acces în secţiunea de administrare.

array

Puteţi să consideraţi un array ca fiind o colecţie de obiecte (matrice). Pentru a înţelege conceptul, să ne amintim de magazinul de casete video de care vă povesteam într-un capitol anterior. Vă spuneam că fiecare film este indexat cu un număr şi astfel ne putem referi la el atunci când cerem filmul vânzătoarei, folosind numărul respectiv.

Să vedem un exemplu:

```
<?
$filme = array("Casablanca",
"Blairwitch Project",
"Matrix","Rambo");
/* Nu putem afişa direct valorile
unui array, puteţi verifica acest
lucru scriind print $filme; şi
rulând scriptul în browser. Putem
însă apela valorile lui, folosind
indexul numeric, care, dacă nu este
definit, începe de la 0. Să cerem
filmul "Casablanca": */
print $filme[0];
/*Pentru a cere "Matrix" scriem:*/
print $filme[2];
/* şi pentru Rambo */
print $filme[3];
?>
```

Indexarea unui array poate fi definită de către programator, aşa cum şi patronul unui magazin de casete video îşi poate numerota casetele după cum îi place.

```
<?
$filme = array(514=>"Casablanca",
      32=>"Blairwitch
      Project",19=>"Matrix",
      1024=>"Rambo");
/* Acum caseta cu Rambo are numărul
1024 şi aşa ne vom şi referi la
ea:*/
print $filme[1024];
/*Pentru a cere "Matrix" scriem:*/
print $filme[19];
/* şi pentru Blairwitch Project */
print $filme[32];
?>
```

Arrayurile pot fi indexate şi asociativ, adică putem folosi stringuri în loc de integers pentru a ne referi la valorile unui array:

```
<?
$filme = array("dragoste"=> "Casa-
blanca", "groaza"=>"Blairwitch
Project", "SF"=>"Matrix",
"actiune"=>"Rambo");
/* Să afişăm "Blairwitch Project"*/
print $filme["groaza"];
/*Pentru a afişa "Matrix" scriem:*/
print $filme["SF"];
/* şi pentru Rambo*/
print $filme["actiune"];
?>
```

Pentru avansaţi: în PHP nu este nevoie să definim întâi variabila sau să-i declarăm tipul. $i = 1 va fi folosit ca integer sau string în funcţie de contextul în care este folosit. Dacă doriţi să forţaţi evaluarea unei variabile ca un anumit tip puteţi folosi conversia de tip, astfel: $i = (int) $i sau $i = (bool) $i. Mai multe detalii despre conversia tipurilor găsiţi în manualul PHP de pe CD, în secţiunea Type casting.

Operatori

Cel mai des întâlnit operator este cel de atribuire, definit prin semnul =. Am observat din exemplele de până acum că dacă scriem $x = 1 nu înseamnă că $x este egal cu 1 ci că i s-a acordat valoarea 1. Deşi la prima vedere egalitatea şi acordarea valorii pot părea a fi acelaşi lucru, ele nu sunt! Să considerăm următoarele variabile:

```
$x = 1;
$y = 7;
```

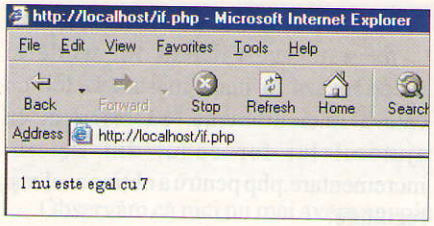
Este $x egal cu $y? Nu este. Valoarea lui $x este 1 în timp ce valoarea lui $y este 7. Folosind operatorul = de atribuire îi putem acorda lui $x valoarea lui $y:

```
$x = 1;
$y = 7;
$x = $y;
```

Abia acum $x este 7 şi este egal cu $y care are şi el valoarea 7. Nu vă lăsaţi înşelaţi de asemănarea cu semnul = din matematică nici măcar atunci când faceţi operaţii complexe. Dacă scriem $rezultat = $x + $y nu înseamnă că $rezultat este egal cu suma celor două, ci că i-am atribuit (acordat) valoarea sumei celor două.

Operatorul de egalitate este == şi se foloseşte cel mai des în propoziţii condiţionale, pentru a testa egalitatea. Opusul său, !=, este operatorul de inegalitate şi se foloseşte în acelaşi scop. Pentru a vă lămuri cum stă treaba cu egalitatea şi atribuirea, testaţi următorul cod:

```
<?
$x = 1; $y = 7;
if($x == $y)
{
      print "$x este egal cu $y";
}
if($x != $y)
{
      print "$x nu este egal cu $y";
}
?>
```



Rulați codul și apoi modificați valorile lui $x și $y cu $x = 13 și $y = 13.

Operatorul de egalitate se folosește pentru a compara egalitatea a două valori. Alți operatori folosiți pentru compararea valorilor variabilelor sunt:

> mai mare
>= mai mare sau egal
< mai mic
<= mai mic sau egal

```
<?
$x = 5;
$y = 4;
if($x > $y)
{
  print "$x este mai mare ca $y";
}
if($x <= $y)
{
  print "$x este mai mic sau egal cu $y";
}
?>
```

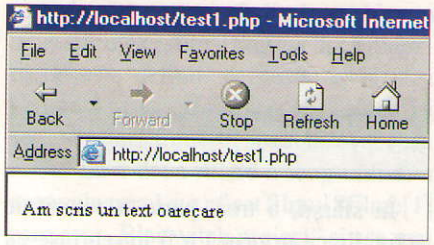
Operatorii pentru stringuri sunt . pentru concatenare și .= pentru atribuirea concatenării. Acești operatori sunt folosiți pentru a uni stringuri, în felul următor:

test1.php

// concatenarea stringurilor

```
$test = 'Am scris un '.'text '.'oarecare';
print $test;
```

// pe ecran va fi afișat „Am scris un text oarecare"

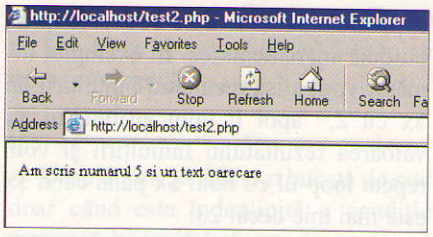


test2.php

// concatenarea stringurilor cu variabile

```
$nr = 5;
print 'Am scris numarul '.$nr.' si
un text '. 'oarecare';
```

// pe ecran va fi afișat „Am scris numărul 5 și un text oarecare"

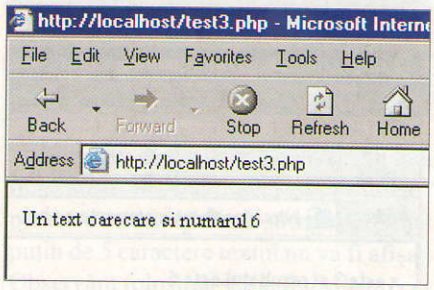


test3.php

// atribuirea concatenării

```
$nr = 6;
$text = "Un text ";
$text .= "oarecare ";
$text .= "și numărul ".$nr;
print $text;
```

// pe ecran va fi afișat „Un text oarecare și numărul 6"



Operatorii logici vă vor veni la îndemână în execuția scriptului atunci când aveți nevoie să lucrați cu valori de adevăr. Vom face un exercițiu de imaginație pentru a explica funcția și utilitatea operatorilor logici. Să presupunem că la intrarea în secțiunea de administrare avem un formular care cere numele și parola de acces în secțiune. Scriptul PHP ar putea verifica aceste informații pentru a autoriza accesul în secțiune folosind operatorii logici astfel:

– operatorul **! (NOT)**

```
if(!parola_e_bună) ... parola nu
  este bună, accesul este interzis
if(!parola_nu_e_bună) ... parola e
  bună, accesul este permis
```

Operatorul ! returnează TRUE dacă valoarea inițială de adevăr e FALSE și FALSE dacă valoarea inițială este TRUE.

– operatorul **|| (OR)**

`if(numele_este_valid || parola este bună) ...` verifică dacă numele sau parola sunt valide și dacă oricare din ele este, returnează valoarea de adevăr TRUE. În acest exemplu de pseudocod dacă numele ar fi valid dar parola nu, i-am acorda utilizatorului acces mai departe, ceea ce nu e de dorit. Trebuie să fim siguri că și numele și parola sunt valide.

Operatorul || returnează TRUE dacă oricare din valorile verificate e TRUE. Returnează FALSE doar dacă amândouă sunt FALSE.

– operatorul **&& (AND)**

`if(numele_este_valid && parola este bună) ...` dacă atât numele cât și parola sunt valide putem acorda utilizatorului acces în secțiunea de administrare.

Operatorul && returnează TRUE doar dacă ambele valori verificate sunt TRUE. El returnează FALSE dacă oricare din ele este FALSE (sau dacă amândouă sunt FALSE).

Structuri de control

Structurile de control sunt instrucțiunile care aduc flexibilitatea în programare și ușurează munca programatorului.

Am văzut cum putem defini o variabilă $x și cum putem obține ca rezultat o altă variabila $total = $x + 1. Putem, cu ajutorul structurilor de control să manipulăm variabilele și rezultatele cu minim de cod. Vom vedea cum, cu ajutorul lui while și for putem folosi aceeași bucată de cod pentru mai multe variabile fără să trebuiască să rescriem codul pentru fiecare valoare a variabilei. Vom învăța și structurile if ... else if ... else și switch cu care vom putea executa cod doar dacă sunt îndeplinite anumite condiții.

while

Structura de control while este folosită pentru a rula același cod pentru mai multe valori ale unei variabile oarecare $x, fără să trebuiască să rescriem codul pentru $x = 1, $x = 2 și așa mai departe pentru fiecare valoare a variabilei. Să folosim pentru testare un cod asemănător celui cu care am pornit la începutul capitolului și să vedem cum structurile de control ne pot ușura munca.

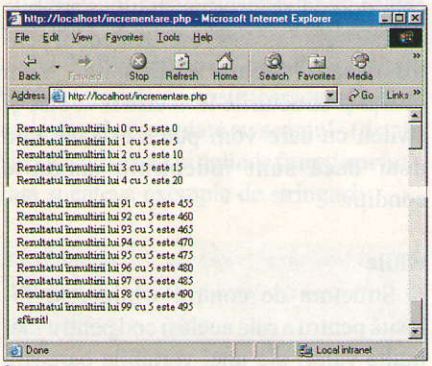
```
<?
$x = 1;
$rezultat = $x*5;
print "Rezultatul înmulțirii lui $x cu
  5 este $rezultat<br>";
$x =2;
$rezultat = $x*5;
print " Rezultatul înmulțirii lui $x cu
  5 este $rezultat<br>";
?>
```

Să presupunem că ar trebui ca pentru toate numerele de la 0 la 99 ar trebui să

calculăm variabila $rezultat şi să afişăm textul „Rezultatul înmulţirii lui $x cu 5 este $rezultat". Dacă ar fi să scriem codul de înmulţire şi afişare pentru fiecare valoare a lui $x de la 0 la 99 ne-ar apuca dimineaţa. Pentru aceasta vom folosi while, cel mai simplu tip de loop şi vom scrie o singură bucată de cod care va prelua automat toate valorile lui $x de la 0 la 99 şi pentru fiecare va înmulţi cu 5 şi va afişa rezultatul automat. Să vedem şi cum:

```
<?
$x = 0;
while ($x < 100)
{
  $rezultat = $x*5;
  print " Rezultatul înmulţirii lui $x
    cu 5 este $rezultat<br>";
  $x++;
}
print "sfârşit!";
?>
```

Salvaţi acest cod într-un fişier incrementare.php şi accesaţi-l din browser pentru a vedea rezultatul.

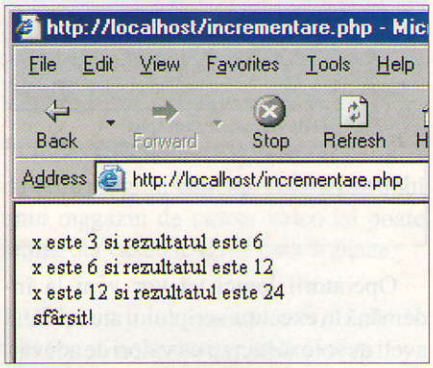


Înmulţirea automată a numerelor de la 0 la 99 cu 5.

Să vedem, pas cu pas, cum funcţionează while. Întâi am definit valoarea iniţială a lui $x. Apoi, pentru fiecare valoare a lui $x mai mică strict decât 100 (0, 1, 2, 3... până când $x ajunge la 99) se calculează variabila $rezultat şi se afişează textul, iar în final $x este incrementat (adunat) cu 1, căpătând astfel o nouă valoare, şi programul o ia din nou de la capăt cu $x = 2, apoi cu $x = 3 şi tot aşa până când $x ajunge la 100. Când $x ajunge la 100 condiţia din while, $x < 100, nu mai este îndeplinită aşa că PHP încetează rularea loop-ului şi continuă cu restul de cod. $++ este o expresie puţin mai specială şi este echivalentă cu incrementarea cu 1. $x++ este astfel prescurtarea pentru $x = $x + 1.

Desigur, putem intreprinde orice acţiune cu $x înainte de a relua loop-ul, nu doar incrementarea. În exemplul următor vom afişa rezultatul înmulţirii lui $x cu 2, apoi îi vom atribui lui $x valoarea rezultatului înmulţirii şi vom repeta loop-ul cu noul $x până când $x este mai mic decât 23:

```
<?
$x = 3;
while ($x < 23)
{
  $rezultat = $x *2;
  print "x este $x şi rezultatul este
    $rezultat <br>";
  $x = $rezultat;
}
print "sfârşit!";
?>
```



Structura for în acţiune!

Iată, în româneşte, ce am făcut: am luat valoarea iniţială a lui $x, 3, şi am început loop-ul. La prima rulare se calculează variabila $rezultat care este $x * 2 = 3 * 2 = 6 şi se afişează pe ecran „x este 3 şi rezultatul este 6". Apoi, folosind operatorul de atribuire „=" în $x = $rezultat îi dăm lui $x valoarea lui $rezultat care este 6 şi loop-ul este reluat de la capăt cu noua valoare $x = 6.

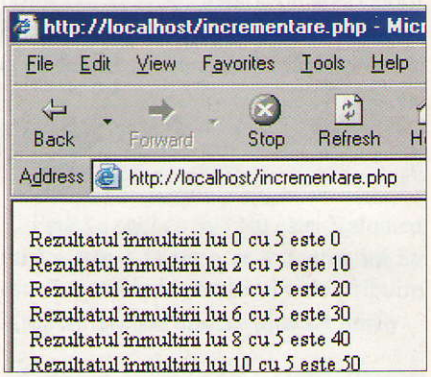
Pentru $x = 6 condiţia $x < 32 este îndeplinită, $rezultat va fi 12 şi, la sfârşit atribuim lui $x această valoare. Pentru $x = 12 valoarea $rezultat va fi 24, valoare atribuită din nou variabilei $x. Cu $x = 24 condiţia pentru rularea loop-ului, $x<23, nu se va îndeplini şi PHP va trece mai departe, la instrucţiunea print "sfârşit!". Vă recomand să testaţi acest cod personal.

Să mai observăm un lucru înainte de a trece mai departe: codul ce urmează a fi executat este încadrat de acolade şi acolada de închidere nu este urmată de punct şi virgulă. Încadrarea în acolade serveşte pentru recunoaşterea bucăţii de cod ce urmează a fi rulată în cazul unei structuri de control. În unele cazuri, vom vedea, acoladele pot fi omise.

for

for este echivalentul lui while şi funcţionează în mare măsură la fel, cu câteva mici diferenţe. Să rescriem cu ajutorul lui for codul din fişierul incrementare.php pentru a obţine aceleaşi rezultate:

```
<?
for ($x = 0; $x < 100; $x++)
{
  $rezultat = $x * 5;
  print " Rezultatul înmulţirii lui $x
    cu 5 este $rezultat<br>";
  $x++;
}
print "sfârşit!";
?>
```



Structura *for* este preferată de programatori deoarece între cele două paranteze ce o precedă sunt conţinute toate instrucţiunile şi condiţiile necesare rulării loop-ului. Astfel, în ($x = 0; $x < 100; $x++), prima parte, $x = 0 reprezintă situaţia iniţială de la care se porneşte rularea loop-ului. A doua parte, $x < 100, reprezintă condiţiile ce trebuie îndeplinite pentru ca loop-ul să fie rulat. Puteţi rula următorul cod ca să observaţi că loop-ul nu va fi rulat deoarece condiţia ca $x să fie mai mic decât 99 nu este îndeplinită.

```
for ($x = 1000; $x < 99; $x++)
{
      print $x;
}
```

În sfârşit, a treia parte o constituie acţiunile ce urmează a fi intreprinse cu variabila în loop. În exemplul nostru variabila este incrementată cu 1 la fiecare rulare.

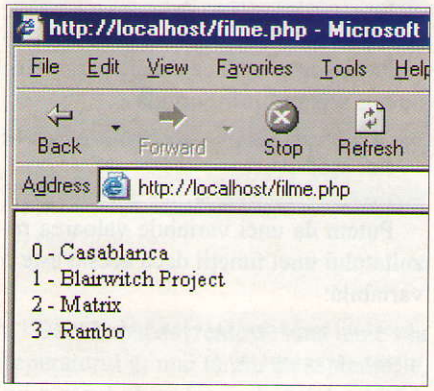
Mai putem adăuga instrucţiuni în această secţiune, separându-le cu virgulă. Iată, spre exemplu, cum am putea scrie toate instrucţiunile pentru manipularea lui $x în ultima secţiune dintre parantezele lui for:

```
<?
for ($x = 0; $x < 100; print "$x<br>",
      $x++);
print "sfarsit!";
?>
```

Observăm că nici nu mai avem nevoie de acolade. Ele nu ne-ar servi la nimic deoarece toate instrucțiunile privind variabila $x se află în a treia parte din paranteze: întâi este afișat $x după care este incrementat și loop-ul o ia de la capăt până când $x ajunge la 100 și condiția de rulare nu mai este îndeplinită.

Putem folosi for pentru afișarea elementelor unui array cu un index numeric ordonat. Să afișăm lista noastră de filme:

```
<?
$filme = array("Casablanca","Blairwitch
  Project","Matrix","Rambo");
$nr_filme = count($filme);
for($i = 0; $i < $nr_filme; $i++)
{
  print "$i - $filme[$i]<br>";
}
?>
```



Rulați scriptul și observați că au fost afișate toate filmele. Cum am făcut acest lucru? Indicându-i lui for de unde să înceapă afișarea elementelor array-ului și unde să se oprească din loop. Ne-am adus aminte că într-un array, dacă indexul nu este definit, pornește de la 0 și pentru a afișa „Casablanca" trebuia să scriem print $filme[0] și am făcut același lucru. La a doua rulare a loop-ului $i a avut valoarea 1 și a afișat filmul $filme[1] care este „Blairwitch project" și așa mai departe pentru fiecare element din array.

Funcția count() este folosită pentru a număra elementele dintr-un array. count($filme) este 4 în cazul de față.

Teme:

1.Adăugați mai multe filme în array și rulați din nou scriptul.

2. Schimbați $i = 0 cu $i = 1 și rulați scriptul.

3. Schimbați $nr_filme=count($filme) cu $nr_filme = count($filme)-1 și rulați din nou scriptul.

if

Dacă dorim să rulăm o bucată de cod doar când este îndeplinită o condiție anume, folosim if. Iată cum, în următorul cod afișăm un text doar dacă numărul de caractere dintr-un string este mai mare decât 5.

```
<?
$text = "Ana are mere";
$nr_caractere = strlen($text);
if($nr_caractere > 5)
{
      print "Textul \"$text\" are mai
            mult de 5 caractere";
}
?>
```

Încercați același cod cu un string mai mic, $text = "Ana" sau $text = "mere" pentru a vedea că dacă stringul are mai puțin de 5 caractere textul nu va fi afișat. Observăm folosirea semnului \ de precedare a caracterelor speciale pentru a afișa ghilimelele în rezultatul de pe ecran, astfel: Textul „Ana are mere" are mai mult de 5 caractere.

Am folosit în acest exemplu funcția strlen care calculează numărul de caractere într-un string. Pentru $text = "Ana are mere", valoarea variabilei $nr_caractere = strlen($text) va fi 12 (spațiile contează). Pentru $text="mere", $nr_caractere va fi 4.

Vă spuneam mai sus că în unele cazuri putem să omitem acoladele. În cazul lui if, dacă partea dintre acolade este o singură linie putem „uita" acoladele și scrie codul asfel:

```
if($nr_caractere > 5)  print "Textul
  \"$text\" are mai mult de 5 caractere";
```

if... else

Ce facem atunci când dorim să afișăm un text dacă o condiție este îndeplinită și alt text dacă aceeași condiție nu este îndeplinită? Folosim if ... else:

```
<?
$text = "mere";
$nr_caractere = strlen($text);
if($nr_caractere > 5)
{
      print "Textul are mai mult de 5
            caractere";
}
else
{
      print "Textul are mai puțin de 5
            caractere";
}
?>
```

Pentru stringul „mere" rezultatul afișat pe ecran va fi „Textul are mai puțin de 5 caractere" și pentru stringul „Ana are mere" rezultatul va fi „Textul are mai mult de 5 caractere". Testați acest lucru rulând codul pentru ambele cazuri.

În construcția if ... else putem omite acoladele dacă instrucțiunile ce urmează a fi executate sunt pe o singură linie:

```
if($nr_caractere > 5) print "Textul
  are mai mult de 5 caractere";
else print "Textul are mai puțin de 5
  caractere";
```

if ... else if ... else

Dar dacă dorim să folosim mai multe condiții și în funcție de fiecare dintre acestea să afișăm altceva? Folosim if ... else if... else și repetăm else if de câte ori avem nevoie.

```
<?
$text = "prune";
$nr_caractere = strlen($text);
if($nr_caractere < 5)
{
      print "Textul are mai puțin de 5
        caractere";
}
else if ($nr_caractere == 5)
{
      print "Textul are exact 5
        caractere";
}
else if($nr_caractere > 12)
{
      print "Textul are mai mult
        de 12 caractere";
}
else
{
      print "Textul are mai puțin de
        5 caractere și mai mult de 12";
}
?>
```

În acest exemplu vedem că else care este folosit chiar la sfârșit, servește pentru toate celelalte cazuri care nu îndeplinesc condițiile specificate în if și else if.

Testați exemplul de mai sus folosind diverse valori pentru $text.

switch

switch este alternativa pentru blocurile condiționale if ... else if ... else. Este preferabil în cazul în care codul dintre acolade este mai mare şi ne-am putea pierde în propoziții condiționale şi paranteze. Să vedem cu un exemplu simplu cum se foloseşte while:

```
<?
$text = "Ana are mere";
$nr_caractere = strlen($text);
switch($nr_caractere)
{
  case 4:
     print "Textul are 4 caractere";
     break;
  case 5:
     print "Textul are 5 caractere";
     break;
  case 12:
     print "Textul are 12 caractere";
     break;
  default:
     print "Textul nu are nici 4 nici
       5 nici 12 caractere";
     break;
}
?>
```

Cazul special „default" de la sfârşit este folosit pentru situația în care dorim să executăm cod şi dacă nici una din condițiile anterioare nu este îndeplinită (fiind astfel similar lui else din structura if ... else if ... else). El poate fi omis şi astfel să executăm cod doar dacă una din condiții este îndeplinită. Instrucțiunea break; trebuie folosită la încheierea fiecărui caz deoarece altfel codul din următorul caz va fi executat. Iată un exemplu:

```
<?
$litera = "a";
switch($litera)
{
  case "a":
    print "Ana ";
  case "b":
    print "Barbu ";
  case "c":
    print "Claudia ";
}
print "este un nume";
?>
```

Rezultatul rulării acestui cod va fi „Ana Barbu Claudia este un nume" şi nu doar „Ana este un nume" aşa cum ne-am fi aşteptat, deoarece în lipsa instrucțiunii break; la încheiera cazurilor rularea codului nu s-a oprit. Mai observăm din acest exemplu că în cadrul structurilor de control, de orice fel nu doar switch, putem folosi stringuri dar va trebui să le încadrăm cu ghilimele (cum nu am fost nevoiți să facem în cazul numerelor). Iată două exemple de folosire a stringurilor într-o structură de control, folosind numere, respectiv stringuri:

```
<?
if ($i == 1)
{
      print "$i este impar";
}
if ($i == 2)
{
      print "$i este par";
}
?>
```

Al doilea exemplu:

```
<?
if($nume == "Ana")
{
      print "Acesta e nume de fată";
}
if ($nume == "Barbu")
{
      print "Acesta este nume de băiat";
}
?>
```

Am învățat până acum ce sunt variabilele şi ce fel de structuri de control avem la dispoziție pentru a lucra cu ele astfel încât să obținem rezultatele dorite cu minim de efort. În continuare vom afla ce sunt funcțiile şi cum acestea ne fac munca chiar mai uşoară.

Funcții

Vă mai amintiți unde am mai întâlnit termenul de funcție? Dacă vă spun că la algebră o să fugiți mâncând pământul? Ei bine, legați-vă de scaun pentru că acolo am întâlnit termenul şi în continuare vă voi explica funcționarea funcțiilor (hmm, asta o fi repetiție?) în PHP. Deşi programarea PHP nu este matematică, modul în care tratează funcțiile este similar. Spre exemplu, la algebră, funcția f(x)=x+1 executa o operație cu x şi dădea un rezultat în funcție de valoarea lui x. Acelaşi lucru îl face şi o funcție PHP.

```
<?
// funcțiile trebuie definite înainte
de a le apela.
function recalculare($x)
{
      $total = $x + 1;
      print $total;
}
$x = 1;
recalculare ($x);
// va afişa pe ecran 2
?>
<br>Putem de asemenea să folosim di-
rect valoarea variabilei ca parametru
(argument) al funcției: <br>
<?
recalculare(16);
// va afişa pe ecran 17
?>
```

Funcțiile pot accepta mai mulți parametri:

```
<?
function inmultire($x, $y)
{
      $rezultat = $x  *  $y;
}
$x = 5;
$y = 6;
print inmultire($y, $y);
?>
```

Putem de asemenea să dăm parametrii direct, separați prin virgulă

```
print inmultire(76,59);
```

sau aşa:

```
print inmultire($x = 76, $y = 59);
```

Putem da unei variabile valoarea rezultatului unei funcții dacă acesta este o variabilă:

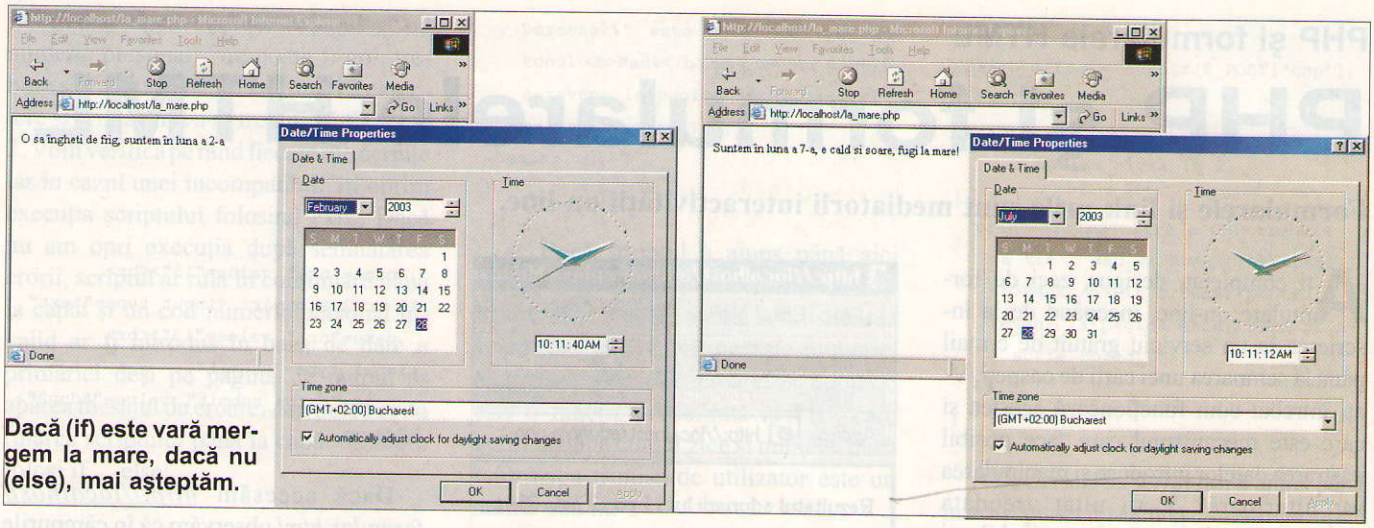
```
$variabila = inmultire($x = 16, $ y = 30);
print $variabila;
```

În pasarea unei variabile către o funcție nu contează numele variabilei ci doar valoarea acesteia, valoare ce urmează a fi prelucrată.

```
<?
function adunare($x)
{
      $rezultat = $x + 1;
}
$a = 5;
print adunare($a);
$b = 10;
print adunare($b);
?>
```

În acest exemplu vedem că doar valoarea variabilei este cea care contează, funcția preluând această valoare.

Funcțiile pot fi definite şi rulate fără a li se specifica vreun argument:



Dacă (if) este vară mergem la mare, dacă nu (else), mai aşteptăm.

```
<?
function vreau_la_mare()
{
 $luna = date("n");
```

/* date("n") returnează reprezentarea numerică a lunii în care suntem, de la 1 pentru ianuarie până la 12 pentru decembrie. Astfel, dacă luna curentă e aprilie, $luna va fi 4 şi dacă e noiembrie $luna va fi 11*/

```
 if($luna < 6 || $luna > 9) print "O să
 îngheți de frig, suntem în luna a
 $luna-a";
 else print "Suntem în luna a $luna-a,
 e cald şi soare, fugi la mare!";
}
vreau_la_mare();
?>
```

Dacă e mai devreme de luna iunie sau (operatorul ||) mai târziu de septembrie, nu e cazul să ne facem bagajele. Altfel, drumul e al nostru!

Puteți modifica (temporar) setările de dată din Windows şi să schimbați luna pentru a vedea ce se întâmpla. Nu uitați să reveniți la setările anterioare!

Am văzut din aceste exemple că rezultatul unei funcții poate fi o valoare (funcția inmultire care oferă rezultatul înmulțirii lui $x cu $y) sau o acțiune (funcția vreau_la_mare() care afişează un text). Rezultatul unei funcții mai poate fi şi o valoare de adevăr, TRUE sau FALSE, ca în exemplul următor:

```
<?
function e_dimineata()
{
 $ora = date("G");
 /*date("G") returnează ora curentă în
 format 0-24 */
 if($ora >= 5 && $ora <=9)
 {
      return TRUE;
 }
 else
 {
      return FALSE;
 }
}

if(e_dimineata()) print "Poți să faci
 cafeaua!";
else print "E prea târziu pentru cafea.";
```

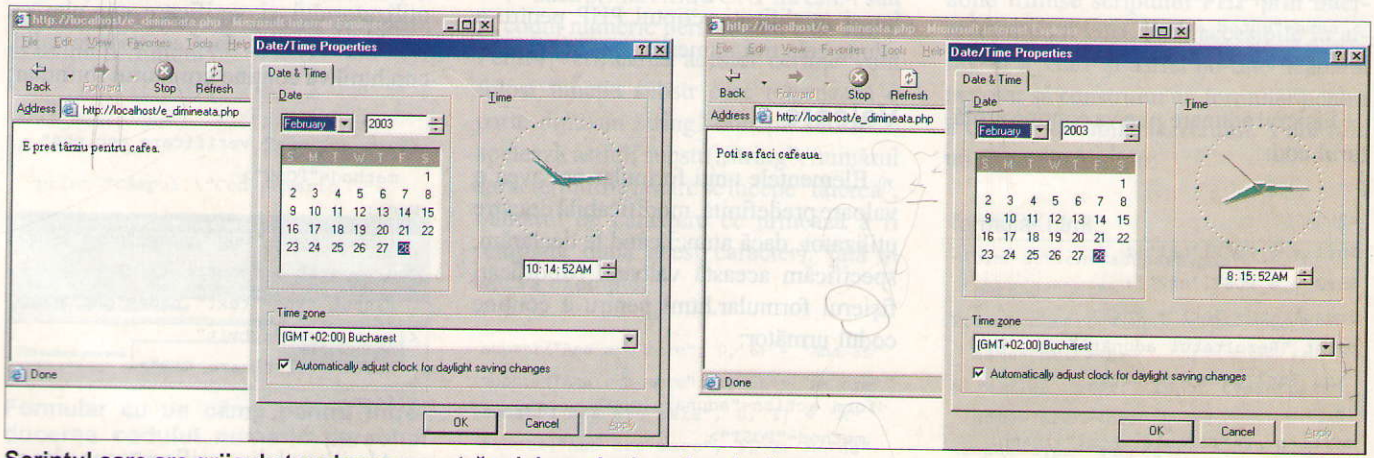
Sau, acelaşi lucru scris folosind operatorul ! NOT:

```
if(!e_dimineata()) print "E prea
 târziu pentru cafea.";
else print "Poți să faci cafeaua.";
?>
```

Această funcție ilustrează foarte bine folosirea operatorilor logici. Dacă ora este mai mare sau egală cu 5 şi mai mică sau egală cu 9, ne putem face cafeaua. Dacă e 4am nu vom bea cafea deşi $ora <=9 şi la fel dacă e ora 22 deşi $ora >=5. Ambele condiții trebuie să fie adevărate pentru ca e_dimineata() să fie TRUE.

Funcții predefinite

Toate funcțiile exemplificate până acum au fost definite de noi. Există însă şi funcții predefinite, integrate în PHP, pentru o mulțime de acțiuni. Un exemplu este funcția date() pe care tocmai am folosit-o şi care returnează ora, luna, anul precum şi alte elemente ale datei. Alt exemplu ar fi funcția count() pe care am folosit-o să numărăm câte elemente sunt într-un array. Există funcții pentru aproape orice în PHP şi vă sfătuiesc să căutați în manualul PHP de pe CD ori de câte ori aveți nevoie de ceva.



Scriptul care are grija de tensiunea noastră: nici un pic de cafea după ora 10.

PHP şi formularele HTML

PHP şi formularele HTML

Formularele şi link-urile sunt mediatorii interactivităţii on-line.

Aţi completat, desigur, zeci de formulare on-line, începând de la înscrierea la un serviciu gratuit de e-mail până la semnarea unei cărţi de oaspeţi. V-aţi întrebat cum funcţionează acestea şi care este mecanismul care face posibil păstrarea datelor introduse şi manipularea lor ulterioară? V-aţi uitat vreodata nedumeriţi la link-urile interminabile şi aparent fără sens din unele pagini de web?

În acest capitol vom afla cum ajunge informaţia dintr-un formular la un script PHP şi cum, un link poate fi mai mult decât ceea ce pare la prima vedere.

Acţiune!

Un formular este compus în mod normal din declaraţia <form...> urmată de elemente de tip <input> şi un buton „submit„. În momentul în care apăsăm butonul submit, datele adăugate de utilizator sunt transmise serverului unde sunt prelucrate.

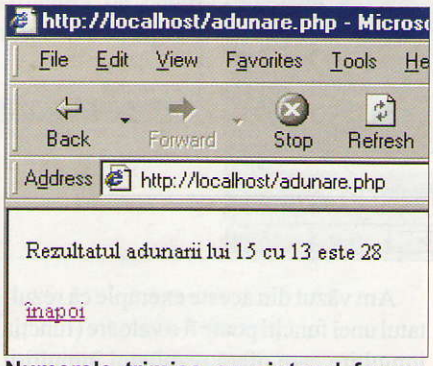
Să vedem practic cum se realizează acest lucru, recreând într-un formular dialogul cu care am început să învăţăm PHP, unde Gigel trebuie să adune două numere. Pentru aceasta vom crea două fişiere, formular.html şi adunare.php.

În fişierul HTML vom avea următorul formular:

```
<form action="adunare.php" method="POST">
Primul număr: <input type="text"
  name="nr1"><br>
Al doilea număr: <input type="text"
  name="nr2"><br>
<input type="submit"
  name="buton_submit" value="Adună">
</form>
```

Fişierul adunare.php va conţine următorul cod:

```
<?
$nr1 = $_POST['nr1'];
$nr2 = $_POST['nr2'];
$rezultat = $nr1 + $nr2;
print "Rezultatul adunării lui $nr1
  cu $nr2 este $rezultat";
?>
<p><a href="formular.html">înapoi
  </a></p>
```



Numerele trimise cu ajutorul formularului au fost adunate şi rezultatul afişat.

Rulaţi *http://localhost/formular.html*, scrieţi câte un număr în fiecare căsuţă de text şi apăsaţi butonul „Adună". Repetaţi experimentul cu alte numere.

Din acest exemplu observăm mai multe lucruri: în declararea tag-ului <form> am specificat fişierul care urmează să prelucreze datele (action="adunare.php"), precum şi metoda prin care se transmit datele către server, în cazul de faţă POST. Elementele formularului au câte un nume după care vor fi recunoscute de către interpretor (ex: $_POST['nr1']). La apăsarea butonului „Adună", informaţia va fi trimisă serverului. $_POST este un array care conţine toate datele trimise prin formular cu metoda POST iar fiecare din elementele acestui array poate fi accesat ca $_POST['nume_variabilă'].

Temă: adăugaţi încă un element, <input type="text" name="nr3"> în formular şi modificaţi scriptul PHP pentru a aduna cele trei numere.

Valoare

Elementele unui formular pot avea o valoare predefinită, modificabilă de către utilizator, dacă atunci când le declarăm, specificăm această valoare. Modificaţi fişierul formular.html pentru a conţine codul următor:

```
<form action="adunare.php"
  method="POST">
Primul număr: <input type="text"
  name="nr1" value="15"><br>
Al doilea număr: <input type="text"
  name="nr2" value="13"><br>
<input type="submit"
  name="buton_submit" value="Adună">
</form>
```

Dacă accesăm *http://localhost/formular.html* observăm că în câmpurile destinate numerelor ce urmează a fi adunate există deja valorile 15 şi respectiv 13. Putem apăsa butonul „Adună" pentru a obţine rezultatul 28 sau putem modifica oricare din valorile prezentate în formular (direct în browser, nu editând codul HTML), pentru a obţine alt rezultat.

Verificare

Un utilizator şugubăţ ar putea introduce un text în loc de un număr sau ar putea lăsa chiar cele două câmpuri goale. În exemplul nostru aceasta n-ar fi o problemă însă dacă am face o aplicaţie serioasă on-line pentru primăria oraşului, cu ajutorul căreia cetăţenii să-şi plătească impozitele, omiterea verificării datelor trimise de utilizator ar duce la haos. Dacă cetăţenii ar putea scrie un text oarecare în loc de codul numeric personal sau nu l-ar completa, fişa de impozit nu ar putea fi procesată.

De aceea, pentru orice fel de date trimise de către utilizatori vom face verificările de rigoare. Pentru a exemplifica, vom face două fişiere noi, cnp.html cu care vom cere codul numeric personal al utilizatorului şi verificare_cnp.php cu care vom verifica dacă acesta este valid. cnp.html va conţine următorul formular:

```
<form action=" verificare_cnp.php"
  method="POST">
Nume:
  <input type="text" name="nume"><br>
Cod numeric personal:
  <input type="text" name="cnp"><br>
<input type="submit"
  value="Verificare CNP">
</form>
```

Algoritmul din verificare_cnp.php va

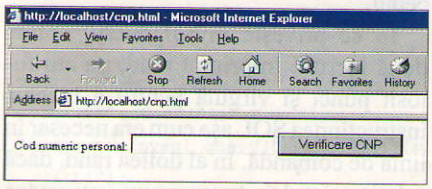
fi orientat după cerințe: câmpul „Cod numeric personal" nu poate fi gol, trebuie să fie numeric, să conțină 13 caractere și prima cifră a sa trebuie să fie 1 sau 2. Vom verifica pe rând fiecare din cerințe iar în cazul unei incompatibilități oprim execuția scriptului folosind exit;. Dacă nu am opri execuția după semnalarea erorii, scriptul ar rula în continuare până la capăt și un cod numeric personal invalid ar fi introdus în baza de date a primăriei deși pe pagina de output ar apărea mesajul de eroare. Am putea evita rularea scriptului până la capăt dacă am folosi if ... else:

```
if cerința 1 nu este îndeplinită
{
  afișează mesaj de eroare
}
else
{
  if cerința 2 nu este îndeplinită
  {
    afișează mesaj de eroare
  }
  else
   {
     if cerința 3 nu este îndeplinită...
   }
}
```

Cu if ... else ne-am pierde în acolade așa că vom folosi exit în verificările noastre pentru a opri rularea ulterioară a scriptului dacă una din cerințe nu este îndeplinită.

În continuare vom scrie scriptul de verificare a codului numeric personal introdus în formularul cnp.html. Scrieți și voi acest cod omițând comentariile ajutătoare.

```
<?
/*dacă utilizatorul nu a introdus nimic
în câmpul "Cod numeric personal", îi
afișăm un mesaj de eroare și oprim
execuția scriptului*/

if($_POST['cnp'] == "")
{
  print "Câmpul \"Cod numeric
    personal\" este gol! Apăsați butonul <b>Back</b> în browser pentru a reveni la pagina anterioară și a-l scrie corect.";
  exit;
}

/* Dacă scriptul a ajuns până aici
înseamnă că cerința anterioară a fost
îndeplinită. Trecem atunci la următoarea
verificare: dacă CNP nu este numeric,
afișăm alt mesaj de eroare. is_numeric
este o funcție predefinită în PHP care
verifică după cum îi zice și numele, dacă
informația trimisă de utilizator este un
număr sau nu*/

if(!is_numeric($_POST['cnp']))
{
  print "Câmpul \"Cod numeric
    personal\" trebuie să fie numeric!
    Apăsați butonul <b>Back</b> în
    browser pentru a reveni la pagina
    anterioară și a-l scrie corect.";
  exit;
}

/* dacă CNP nu are exact 13 caractere,
afișăm un mesaj de eroare. Funcția strlen
returnează numărul de caractere (inclusiv
spații) dintr-un string. strlen("mere")=4,
strlen("120")=3 și strlen("4 mere")=6.*/

$nr_caractere = strlen($_POST['cnp']);
if($nr_caractere != 13)
{
  print "Câmpul \"Cod numeric
    personal\" trebuie să aibă exact 13
    caractere! Apăsați butonul <b>Back
    </b> în browser pentru a reveni la
    pagina anterioară și a-l scrie
    corect.";
  exit;
}

/* dacă prima cifră a CNP nu este 1 sau
2, codul numeric personal nu este valid.
Pentru verificarea acestei cerințe vom
folosi funcția substr care returnează o
parte dintr-un string. Funcția substr se
apelează astfel: substr(„string", numărul
caracterului de la care se începe "tăierea",
numărul de caractere ce urmează a fi
returnate după acest caracter). Iată și
câteva exemple:

substr("Ana are mere", 0, 6) = "Ana ar"
substr("Ana are mere", 1, 6) = "na are"
substr("Ana are mere", 0, 1) = "A"
substr("Ana are mere", 1, 2) = "na"
substr("Ana are mere", 6, 5) = "e mere"
*/
$prima_cifra = substr($_POST['cnp'],
  0, 1);
if($prima_cifra != 1 &&
      $prima_cifra != 2)
{
  print "Prima cifra a CNP trebuie
    să fie 1 sau 2! Apăsați butonul
    <b>Back</b> în browser pentru a
    reveni la pagina anterioară și a-
    l scrie corect.";
  exit;
}

/*Dacă rularea scriptului a ajuns până
în acest punct înseamnă că toate cerințele
au fost îndeplinite și codul numeric personal este corect. În aplicația noastră ar
urma să îl introducem în baza de date dar
cum nu acesta era scopul exemplului,
vom afișa doar un mesaj de confirmare:*/

print "Acesta este un CNP valid!";
?>
```



Formular cu un câmp pentru introducerea codului numeric personal pentru verificare.

GET

Formularele pot fi transmise serverului și cu metoda GET (<form action="fișier.php" method="GET">) însă este preferabil să evitați să folosiți GET în formulare din trei motive: datele din cadrul formularului sunt transmise serverului prin atașarea la URL, nu se pot transmite decât caractere ASCII și cantitatea de informație transmisibilă este limitată ca mărime.

Dacă folosim metoda GET trebuie ca și în cadrul scriptului PHP să accesăm variabila ca atare (ex: $_GET['cnp'] în loc de $_POST['cnp']). Datele formularului sunt transmise prin URL sub forma http://www.adresa.site/fișier.php?variabila1=valoare1 &variabila2=valoare2. Ca regulă generală, orice variabile trimise scriptului PHP prin intermediul unui URL sunt accesibile în array-ul $_GET și astfel nu este neapărat necesar să construim un formular pentru a trimite variabile serverului. Vom rula un exemplu simplu:

formular.html

```
<form action="adunare.php"
  method="GET">
Primul număr: <input type="text"
  name="nr1"><br>
Al doilea număr: <input type="text"
  name="nr2"><br>
<input type="submit">
```

Variabilele transmise prin metoda GET sunt vizibile în bara de adrese a browserului.

```
</form>
```

adunare.php

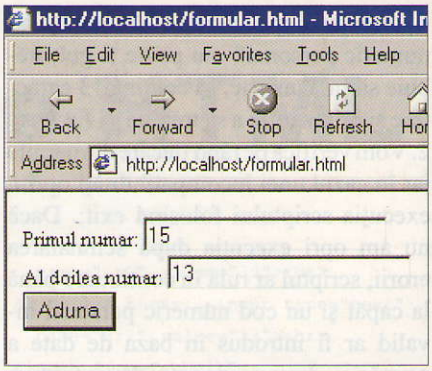
```
<?
$nr1 = $_GET['nr1'];
$nr2 = $_GET['nr2'];
$rezultat = $nr1 + $nr2;
print "Rezultatul adunării lui $nr1
  cu $nr2 este $rezultat";
?>
```

Rulând acest exemplu putem vedea cum numele şi valorile elementelor formularului se transmit fişierului PHP în URL. Putem accesa direct în browser http://localhost/adunare.php?nr1=9053&nr2=4906 sau http://localhost/adunare.php?nr1=423&nr2=546, transmiţând datele prin URL, fără să completăm formularul.

În cadrul site-ului nostru vom folosi formulare pentru a lăsa posibilitatea utilizatorilor să adauge comentarii şi pentru a trimite adresa unde doresc să primească cărţile cumpărate. Datele din formularele completate de către vizitator vor fi trimise serverului prin metoda POST. Tot în cadrul site-ului vom folosi de multe ori GET pentru a obţine variabile dintr-un URL. Spre exemplu, pentru a vedea o carte anume adresa va fi http://localhost/carte.php?carte_id=345. Scriptul PHP va selecta din baza de date doar informaţiile despre cartea cu id 345 şi le va afişa. Vom vedea cum în următoarele două capitole.

Globals

Există o setare în php.ini care v-ar permite accesul la variabilele trimise de utilizator fără să le accesaţi din array. Astfel, într-un form trimis prin metoda POST, valoarea din <input type="text" nume="oarecare"> ar putea fi accesibilă ca $oarecare în loc de $_POST['oarecare']. Acelaşi lucru este valabil şi pentru metoda GET sau pentru elementele din array-ul $_FILES. Variabila $_GET['oarecare'] ar putea fi accesibilă ca $oarecare iar $_FILES['fisier']['size'] ca $fisier_size. Deşi poate părea mai uşor de programat aşa, este recomandabil să lăsaţi setarea Globals=Off în php.ini (aşa cum este setată implicit) şi întotdeauna să accesaţi variabilele din array-ul din care provin. Asfel evitaţi o serie de probleme de securitate, probleme pe care le vom explica mai pe larg în capitolul dedicat securităţii aplicaţiilor web). ■



Formularul cu ajutorul căruia vom trimite două numere pentru a fi calculate la server.

Lucrul cu baze de date

PHP şi MySQL

Dezvoltarea explozivă a limbajului PHP din ultimii ani este în cea mai mare parte datorată uşurinţei cu care acesta lucrează cu bazele de date.

PHP oferă programatorului o mulţime de funcţii predefinite pentru lucrul cu baza de date. Aceasta înseamnă că din cadrul PHP putem executa toate operaţiunile pe care le-am făcut în capitolul dedicat învăţării MySQL. Să ne reamintim paşii pe care i-am facut pentru a interacţiona cu baza de date din linia de comandă:

1. ne-am conectat cu numele şi parola (mysql -u root -p)
2. am ales baza de date cu care să interacţionăm (USE librarie)
3. am executat o comandă SQL (SELECT * FROM carti;)

Din PHP putem face acelaşi lucru, cu ajutorul funcţiilor predefinite. Să facem primul nostru script cu care să interacţionăm cu baza de date (nu uitaţi că serverul MySQL trebuie să fie pornit!):

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
mysql_query("SELECT * FROM carti");
?>
```

Observăm similitudinea între cele două abordări. Cu funcţia mysql_connect ne conectăm la baza de date specificând hostul la care aceasta se află (localhost), numele de utilizator (root) şi parola (pentru că aceasta este goală folosim un string gol: ""). Funcţia mysql_select_db ("librarie") este echivalentă cu USE librarie. Aceste două funcţii trebuie apelate întotdeauna atunci când dorim să lucrăm cu baza de date, înainte de a efectua vreo interogare. Putem efectua oricât de multe interogări dorim fără să fie nevoie să ne reconectăm pentru fiecare din ele la baza de date, doar prin apelarea mysql_connect şi mysql_select_db o singură dată la început.

În ce priveşte mysql_query, notăm două lucruri. În primul rând, nu am folosit punct şi virgulă pentru a încheia instrucţiunea SQL aşa cum era necesar în linia de comandă. În al doilea rând, dacă rulaţi scriptul în browser nu veţi vedea nimic pe ecran aşa cum probabil v-aţi aşteptat. Aceasta pentru că mysql_query

execută interogarea dar nu afişează rezultatul ci returnează o valoare: TRUE dacă interogarea a fost efectuată cu succes sau FALSE dacă aceasta a eşuat. Pentru instrucţiunile de tip SELECT, SHOW, EXPLAIN sau DESCRIBE, mysql_query returnează şi un identificator de resurse pe care îl putem atribui unei variabile.

Resursele sunt variabile speciale care conţin referinţe către resurse externe (precum rezultatul unei interogări a bazei de date) şi pot fi manipulate cu ajutorul funcţiilor.

Este evident că dacă rezultatul interogării ar fi fost afişat direct în browser, nu am putea modifica aspectul sau funcţia paginilor de web dinamice şi amazon.com ar fi o simplă înşiruire de date.

Să ne convingem că rezultatul interogării este o resursă. Pentru aceasta modificaţi ultima linie a scriptului şi scrieţi:

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$resursa = mysql_query("SELECT * FROM
  carti");
print $resursa;
?>
```

Putem de asemenea scrie interogarea SQL într-o variabilă pe care să o folosim ca parametru al funcţiei mysql_query. Această metodă este preferată de mulţi deoarece textul interogării este mai uşor de reperat în cadrul scriptului:

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "SELECT * FROM carti";
$resursa = mysql_query($sql);
print $resursa;
?>
```

PHP oferă funcţii pentru accesarea resurselor care rezultă din mysql_query. Cu ajutorul mysql_num_rows putem afla câte rânduri a retunat interogarea:

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librărie");
$resursa = mysql_query("SELECT * FROM
  carti");
$nr = mysql_num_rows($resursa);
print "Sunt $nr cărţi în baza de date";
?>
```

Pentru a afişa valori cadrul din resursei returnate putem folosi mysql_result. În exemplul următor afişăm conţinutul câmpului titlu de pe primul rând al tabelului (numerotarea începe de la 0):

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$resursa = mysql_query("SELECT * FROM
  carti");
$rezultat = mysql_result($resursa, 0,
  "titlu");
print $rezultat;
?>
```

Parametrii funcţiei mysql_result sunt: resursa ($resursa), rândul (0, primul rând al tabelului) şi numele câmpului ("titlu");

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
/* afişează conţinutul câmpului de-
scriere de pe cel de-al doilea rând al
tabelului */
$resursa = mysql_query("SELECT * FROM
  carti");
$rezultat = mysql_result($resursa, 1,
  "descriere");
print $rezultat;
/* sau conţinutul câmpului id_carte de
pe cel de-al treilea rând al tabelului*/
print mysql_result($resursa, 2,
  "id_carte");
?>
```

Mysql_result este greoi de folosit deoarece suntem nevoiţi să accesăm fiecare coloană a fiecărui rând în parte. În ajutorul nostru vine mysql_fetch_array cu care putem accesa valorile din tabelul returnat în interogare dintr-un array. Putem accesa resursa ca un array numeric:

test.php

```
<?
mysql_connect("localhost","root","");
mysql_select_db("librarie");
$resursa = mysql_query("SELECT *
  FROM carti");
$arrRezultat =
  mysql_fetch_array($resursa,MYSQL_NUM);
print_r ($arrRezultat);
?>
```

sau asociativ:

```
$resursa = mysql_query("SELECT *
  FROM carti");
$arrRezultat =
  mysql_fetch_array($resursa,MYSQL_ASSOC);
print_r ($arrRezultat);
```

Dacă nu specificăm tipul array-ului sau folosim MYSQL_BOTH, vom putea accesa rezultatul atât numeric cât şi asociativ.

```
$resursa = mysql_query("SELECT *
  FROM carti");
$arrRezultat =
  mysql_fetch_array($resursa);
print $arrRezultat[1]."
  ".$arrRezultat[3]."<br>";
print $arrRezultat['id_carte']."
  ".$arrRezultat['titlu'];
```

Observăm folosirea operatorului . de concatenare a stringurilor pentru a interpune spaţii şi line break între variabilele ce urmează a fi afişate. Mai observăm că, deşi mysql_num_rows ne spune că avem mai multe cărţi, print_r ($arrRezultat) sau print $arrRezultat['titlu'] ne afişează doar primul element al array-ului. Putem afişa toate valorile array-ului folosind while:

```
$resursa = mysql_query("SELECT *
  FROM carti");
while($row=mysql_fetch_array($resursa))
{
  print $row['titlu']." "
    .$row['descriere']."<br>";
}
```

Folosind while scriptul trece prin toate valorile array-ului până la sfârşit şi pentru fiecare din ele afişează valorile câmpurilor titlu şi descriere. Variabila $row este la rândul ei un array (observaţi că am apelat-o ca atare, $row['titlu']) şi asta pentru că $arrRezultat este un array multidimensional.

Ce este un array multidimensional? Un array care conţine la rândul lui alte array-uri. În cazul de faţă, $arrRezultat este mulţimea array-urilor de rânduri din tabel iar fiecare $row este un array care cuprinde valorile rândului respectiv. Spre exemplu, în pseudocod:

```
$arrRezultat = array(rândul 1,
  rândul 2, rândul 3);
$rândul1 = array(titlu de pe rândul 1,
  descriere de pe rândul 1);
$rândul2 = array(titlu de pe rândul 2,
  descriere de pe rândul 2);
$rândul3 = array(titlu de pe rândul 3,
  descriere de pe rândul 3);
```

După cum vom vedea în continuare, aceste câteva funcţii ne vor fi de ajuns pentru a putea construi un site dinamic interactiv şi rareori veţi avea nevoie de mai mult. Vă recomand ca de câte ori sunteţi în impas să consultaţi manualul PHP pentru a afla informaţii despre celelalte funcţii MySQL disponibile.

Mai puţin înseamnă mai mult

Să revenim la site-ul nostru. Dacă ar fi fost să facem site-ul în HTML, pentru fiecare carte ar fi trebuit să concepem câte o pagină şi pentru fiecare nouă carte ar fi trebuit să modificăm pagina domeniului căruia îi aparţine. Cu PHP în schimb tot ce trebuie să facem este prima pagină, o pagină pentru domeniu şi una pentru carte. Atât - trei pagini vor fi tot ce avem nevoie, indiferent de numărul cărţilor sau domeniilor din baza de date. În continuare vom exersa funcţiile MySQL ale PHP creând o schiţă a site-ului.

Spuneam că pe prima pagină vom avea numele de domenii. Numele de domenii le obţinem folosind interogarea SELECT nume_domeniu from domenii iar pentru a le afişa, scriptul PHP pentru prima pagină va arăta ca în exemplul următor:

index.php

```
<html>
<head>
<meta  http-equiv="Content-Type"
content="text/html; charset=iso-8859-2">
<title>librăria mea</title>
</head>
<body>
<h1>librăria mea</h1>
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
/* să afişăm pe prima pagină numărul de
cărţi din baza de date: */
$sql1 = "SELECT * FROM carti";
$resursa1 = mysql_query($sql1);
$nr = mysql_num_rows($resursa1);
print "<p>Sunt $nr carti în baza de
  date</p>";
?>
<h2>Domenii</h2>
<?
/* şi în continuare afişăm numele de
domenii: */
$sql2 = "SELECT nume_domeniu
  FROM domenii";
$resursa2 = mysql_query($sql2);
while($row=mysql_fetch_array($resursa2))
{
```

O primă pagină ce conţine lista domeniilor disponibile.

```
  print $row['nume_domeniu'].'<br>';
}
?>
</body></html>
```

Scrieţi şi rulaţi acest script. E interesant, spuneţi, dar nu suficient de impresionant? Ei bine, atunci să facem în aşa fel încât să putem da click pe fiecare nume de domeniu şi să intrăm într-o pagină unde ne sunt afişate cărţile din domeniul respectiv.

Cum facem acest lucru? Ne reamintim că am legat tabelele domenii şi carti prin id_domeniu. În limbaj SQL am putea să folosim interogarea `SELECT titlu from carti, domenii where nume_domeniu='Poezii' and domenii.id_domeniu=carti.id_domeniu;` pentru a afişa toate titlurile din categoria „Poezii" şi la fel pentru toate celelalte categorii. Nu vom face câte o pagină pentru fiecare domeniu în parte deoarece acesta este exact scopul nostru: să scriem un singur script care să afişeze dinamic rezultatele: cărţile de poezii pentru domeniul „Poezii" şi operele de teatru pentru domeniul „Teatru". Pentru aceasta vom folosi o variabilă care să conţină numele domeniului, variabilă pe care o transmitem scriptului domeniu.php printr-un URL. Va trebui să modificăm în index.php linia

```
print $row['nume_domeniu'].'<br>';
```

şi să transformăm numele de domeniu într-un link prin care să transmitem variabila scriptului din domeniu.php.

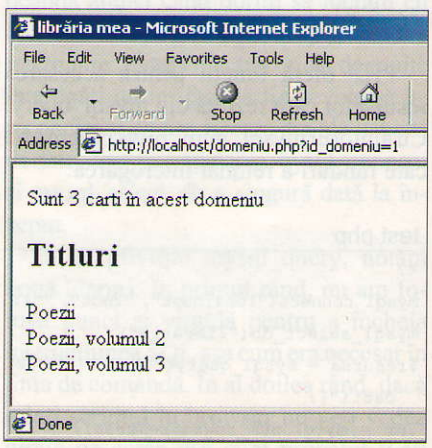
```
print '<a href="domeniu.php?
  nume_domeniu='.$row['nume_domeniu'].'
  ">'.$row['nume_domeniu'].'</a><br>';
```

Astfel în prima pagină toate numele de domenii vor fi link-uri, ca de exemplu Poezie. Fiţi atenţi la dispunerea ghilimelelor în cadrul scriptului, în special la modul în care am inclus ghilimele duble într-un string încadrat de ghilimele simple.

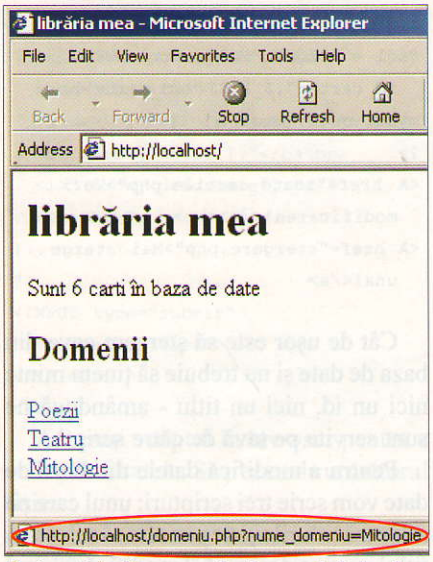
Să facem în continuare scriptul care afişează titlurile disponibile în domeniul pe care am dat click. În fişierul domeniu.php vom folosi variabila trimisă prin URL pentru a obţine din baza de date doar cărţile aparţinând domeniului respectiv:

domeniu.php

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-2">

<title>librăria mea</title>
</head>
<body>
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "SELECT titlu FROM carti,
  domenii WHERE nume_domeniu=
  '".$_GET['nume_domeniu']."' AND
  domenii.id_domeniu=carti.id_domeniu";
$resursa = mysql_query($sql);
$nr = mysql_num_rows($resursa);
print "<p>Sunt $nr cărţi în acest
  domeniu</p>";
?>
<h2>Titluri</h2>
<?

while($row=mysql_fetch_array($resursa))
```



Lista cărţilor din domeniul Poezii alese după id_domeniu.



Putem trimite numele domeniului prin URL.

```
{
  print $row['titlu'].'<br>';
}
?>
</body>
</html>
```

Vă spuneam în capitolul despre MySQL cum id-urile ne pot uşura munca. Acest lucru devine evident atunci când creem pagini de web dinamice deoarece sintaxa interogărilor SQL poate fi simplificată foarte mult.

Fiecare nume de domeniu din tabelul domenii are câte un id asociat şi atunci, ştiind că domeniul Poezii are id_domeniu 1 am putea alege cărţile din domeniul respectiv folosind simplu: SELECT * FROM carti WHERE id_domeniu=1 în loc de mai greu de intelesul SELECT titlu FROM carti, domenii WHERE nume_domeniu='Poezii' AND domenii.id_domeniu=carti.id_domeniu. Cum? Trimiţând valoarea lui id_domeniu prin URL, pentru fiecare domeniu în parte. Modificaţi index.php astfel:

```
$sql2 = "SELECT id_domeniu,
  nume_domeniu FROM domenii";
$resursa2 = mysql_query($sql2);
while($row =
  mysql_fetch_array($resursa2))
{
  print '<a href="domeniu.php?
    id_domeniu=.$row['id_domeniu'].'">
    '.$row['nume_domeniu'].'</a><br>';
}
```

Acum link-urile către pagina domenii.php vor fi de tipul Poezii. Să modificăm acum şi interogarea $sql din domeniu.php şi să o simplificăm folosind id_domeniu ca referinţă:

```
$sql = "SELECT titlu FROM carti WHERE
  id_domeniu=".$_GET['id_domeniu'];
```

Putem, în mod similar, face o pagină care să preia id_carte din URL şi să afişeze pentru cartea respectivă titlul, autorul şi descrierea. Vom face acest lucru într-unul din capitolele următoare. Pentru moment însă, este suficient să înţelegeţi funcţiile MySQL şi modul în care se folosesc variabilele în interogările SQL.

Şi chiar mai mult

Putem folosi funcţia mysql_query pentru a efectua şi alte operaţii cu baza de date: INSERT, UPDATE sau DELETE. Să facem întâi un script care ne afişează toate titlurile din tabelul carti, script de care vom avea nevoie în continuare pentru a consulta baza de date fără să fim nevoiţi să intrăm în linia de comandă. Vom prezenta datele într-un tabel HTML:

toate_cartile.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "SELECT id_carte, titlu FROM
  carti";
$resursa = mysql_query($sql);
// deschidem tabelul:
print "<table border='1'>";
// scriem fiecare rezultat pe un rând:
while($row=mysql_fetch_array($resursa))
{
  print
    "<tr>
      <td>".$row['id_carte']."</td>
      <td>".$row['titlu']."</td>
    </tr>";
}
// şi închidem tabelul
print "</table>";
?>
```

Rulaţi scriptul accesând adresa http://localhost/toate_cartile.php şi apoi creaţi un nou fişier, operatii.php:

operaţii.php

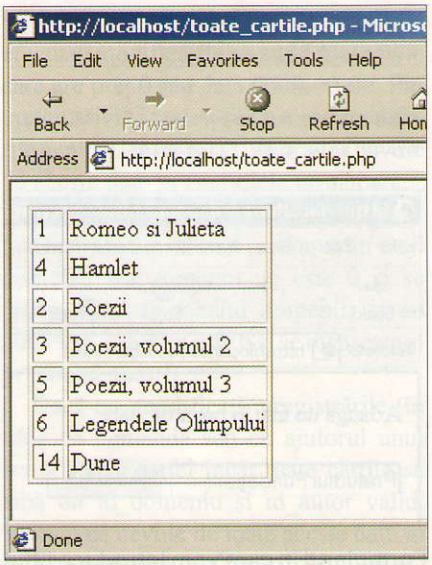
```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "INSERT INTO carti(titlu)
  VALUES('Dune')";
mysql_query($sql);
?>
```

Putem trimite id-ul domeniului, pentru o orientare mai uşoară.

```
<A href="toate_cartile.php">
  Vezi modificarea!</a>
```

Rulaţi scriptul în browser apoi accesaţi http://localhost/toate_cartile.php pentru a vedea că noul titlu a fost introdus în tabel şi i-a fost acordat automat şi un id. Dacă folosiţi Opera sau setarea de refresh din Internet Options din Internet Explorer „Check for newer versions of stored pages - Never", va fi nevoie să reîncărcaţi pagina (refresh).

Să modificăm operaţii.php pentru a vedea cum folosim instrucţiunea DELETE, rescriind variabila $sql:



Toate cărţile din baza de date.

operații.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "DELETE FROM carti WHERE
  titlu='Dune'";
mysql_query($sql);
?>
<A href="toate_cartile.php">Vezi
  modificarea!</a>
```

Rulați și acest exemplu și apoi accesați http://localhost/toate_cartile.php pentru a vedea că înregistrarea a fost ștearsă din tabel.

PHP, MySQL și formularele

A sosit momentul să punem cap la cap toate lucrurile învățate până acum.

Folosind formularele HTML putem interacționa cu baza de date fără să trecem prin chinurile scrierii interogării SQL în linia de comandă sau în cadrul unui script PHP ori de câte ori vrem să adăugăm, să modificăm sau să ștergem vreo înregistrare.

Să facem un formular care să ne ajute în adăugarea de titluri noi:

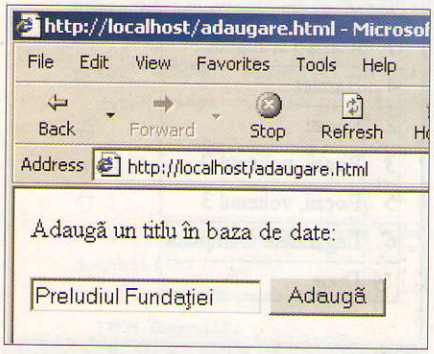
adaugare.html

```
Adaugă un titlu în baza de date:
<form action="adaugare.php"
  method="POST">
<input type="text" name="titlu">
<INPUT type="submit" value="Adaugă">
</form>
```

și iată și scriptul care va prelua valoarea din <input type="text" name="titlu"> transmisă prin metoda POST și o va adăuga în baza de date:

adaugare.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "INSERT INTO carti(titlu)
  VALUES('".$_POST['titlu']."')";
mysql_query($sql);
?>
<A href="toate_cartile.php">Vezi
  modificarea!</a><br>
<A href="adaugare.html">Adaugă încă un
  titlu!</a>
```



Formularul în care vom introduce titlul ce urmează a fi inclus în baza de date.

Accesați http://localhost/adaugare.html și puteți adăuga oricâte titluri doriți! Vă puteți verifica oricând, accesând http://localhost/toate_cartile.php.

Să scriem acum un script cu care să ștergem titluri din baza de date și cu această ocazie vom vedea și cum folosim alte controale de formulare decât <input type="text"> sau <input type="submit">.

În exemplul următor vom vedea cum afișăm titlurile într-un dropdown list pentru a putea selecta titlul care urmează să-l ștergem.

Vom folosi un script PHP pentru a afișa utilizatorului o listă cu titlurile disponibile în baza de date, urmând ca el să aleagă titlul ce urmează a fi șters.

stergere.php

```
Sterge un titlu din baza de date:
<form action="stergere_act.php"
  method="POST">
<select name="id_carte">
<?
mysql_connect("localhost","root","");
mysql_select_db("librarie");
$sql = "SELECT id_carte, titlu FROM
  carti ORDER BY titlu ASC";
$resursa = mysql_query($sql);
while($row =
 mysql_fetch_array($resursa))
{
  print "<option
    value='".$row['id_carte']."'>
    ".$row['titlu']."</option>";
}
?>
</select>
<INPUT type="submit" value="Sterge">
</form>
```

În acest formular transmitem variabila cu numele id_carte (definită în <select name="id_carte">) a cărei valoare este conținută în tagul option (<option value='".$row['id_carte']."'>). Iată și scriptul care preia această variabilă și șterge rândul corespondent din baza de date:

stergere_act.php

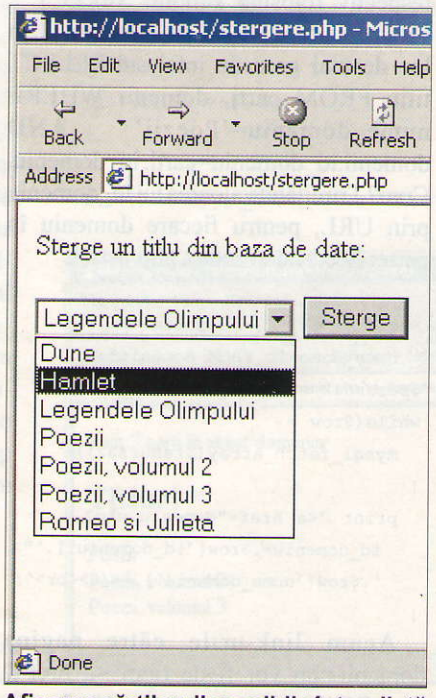
```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "DELETE FROM carti WHERE
  id_carte=".$_POST['id_carte'];
mysql_query($sql);
?>
<A href="toate_cartile.php">Vezi
  modificarea!</a><br>
<A href="stergere.php">Mai sterge
  una!</a>
```

Cât de ușor este să ștergem ceva din baza de date și nu trebuie să ținem minte nici un id, nici un titlu - amândouă ne sunt servite pe tavă de către script!

Pentru a modifica datele din baza de date vom scrie trei scripturi: unul care ne afișează titlurile și ne permite să selectăm titlul pe care dorim să îl modificăm, unul care să afișeze datele într-un formular unde să le putem modifica și încă unul care să trimită datele modificate la baza de date:

lista_carti.php

```
Alege una din carti si apasa butonul
Modifica:
<FORM action="modifica.php"
  method="POST">
<?
mysql_connect("localhost","root","");
mysql_select_db("librarie");
$sql = "SELECT id_carte, titlu,
  descriere FROM carti";
$resursa = mysql_query($sql);
while($row =
  mysql_fetch_array($resursa))
```



Afișarea cărților disponibile într-o listă dropdown pentru o selecție rapidă.

```
{
  print "<input type='radio'
  name='id_carte'
  value='".$row['id_carte']."'>
  <b>".$row['titlu']."</b><br>
  <i>".$row['descriere']."</i>
  <br><br>";
}
?>
<INPUT type="submit"
  value="Modifică">
</form>
```

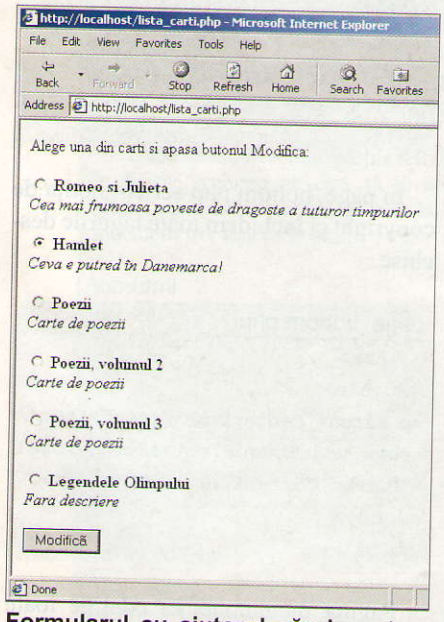
Accesați scriptul în browser pentru a vedea rezultatele. Să trecem la următorul fișier:

modifica.php

```
Modifica aceasta carte:
<FORM action="modifica_act.php"
  method="POST">
<?
```

/* Luăm titlul și descrierea din nou din baza de date pentru a le afișa în controalele formularului deoarece din fișierul precedent nu am primit decât o singură variabilă, $_POST['id_carte'], din butonul radio (<input type='radio' name='id_carte' value='".$row['id_carte']."'>)*/

```
mysql_connect("localhost","root","");
mysql_select_db("librarie");
$sql = "SELECT titlu, descriere
  FROM carti WHERE
  id_carte=".$_POST['id_carte'];
$resursa = mysql_query($sql);
```



Formularul cu ajutorul căruia putem alege cartea ale cărei date dorim să le modificăm.

```
while($row =
  mysql_fetch_array($resursa))
{
  print '<input type="text"
    name="titlu"
    value="'.$row['titlu'].'"><br>
  <textarea name="descriere">'
    .$row['descriere'].
    '</textarea><br>';
}
```

/* și folosind un control ascuns transmitem mai departe către modifica_act.php id-ul cărții deoarece va avea nevoie de el atunci când va opera efectiv modificarea în baza de date */

```
print '<input type="hidden"
  name="id_carte"
  value="'.$_POST['id_carte'].'">';
?>
<INPUT type="submit"
  value="Modifică">
</form>
```

În fine, ultimul script din această serie, modifică_act.php preia variabilele $_POST['titlu'] și $_POST['descriere'] din formularul precedent și actualizează înregistrarea cărții a cărui id_carte are valoarea lui $_POST['id_carte'].

La sfârșitul scriptului folosim header() pentru a redirecționa automat browserul către lista_carti.php după ce modificarea a fost efectuată:

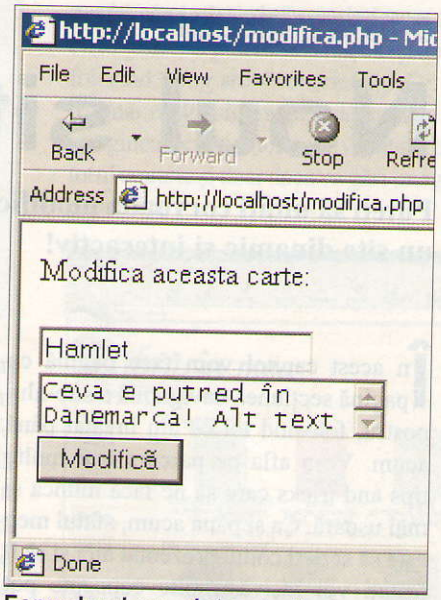
modifica_act.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "UPDATE carti SET
  titlu='".$_POST['titlu']."',
  descriere='".$_POST['descriere']."'
  WHERE id_carte=".$_POST['id_carte'];
mysql_query($sql);
header("location: lista_carti.php");
```

Accesați acum adresa http://localhost/lista_carti.php și modificați după plac orice titlu sau descriere a unei cărți! De-acum puteți spune Adio! lucrului cu baza de date din linia de comandă.

În acest capitol am văzut cum putem face pagini dinamice și interactive și cât de ușor este să creezi și să ții „la zi" un site cu un număr uriaș de pagini folosind doar câteva scripturi.

Ca începători în programare v-ați putea simți frustrați de complexitatea ghilimelelor, variabilelor sau array-urilor dar nu renunțați, aveți nevoie doar de ceva exercițiu și în curând toate vi se vor părea



Formularul cu ajutorul căruia modificăm efectiv detaliile cărții.

floare la ureche. Nu uitați: de câte ori lucrurile nu funcționează așa cum ar trebui, verificați dacă ați pus punct și virgulă la sfârșitul propozițiilor, dacă nu ați scris greșit numele unei variabile sau funcții sau dacă ați pus toate ghilimelele la locul lor. Înainte de a trece la capitolul următor vă recomand să faceți câteva exerciții.

Noi avem câteva cărți în baza de date însă nici una nu are preț sau dată (de fapt valorile sunt 0 pentru preț lși 0000-00-00 pentru dată).

Scrieți scripturi pentru a modifica prețul și data fiecărei cărți, la fel ca în ultimul exemplu prezentat. Nu uitați că prețul este de tip INT în baza de date și va trebui să îl scrieți ca „98500" nu „98,500", „98500 lei" sau „98,500". La fel, data este în format yyyy-mm-dd (ex: 2003-02-23).

După ce ați scris scripturile, rulațile și efectuați modificările pentru fiecare carte care are preț 0 sau data 0000-00-00. Fiți imaginativi și scrieți prețuri și date diferite pentru fiecare carte. Vom avea nevoie de aceste date în capitolele următoare.

Înainte de a continua, verificați dacă toate cărțile din tabelul carti aparțin unui domeniu (id_domeniu nu este 0 și se regăsește și în tabelul domenii) și au autor (id_autor nu este 0 și se regăsește și în tabelul autori).

Dacă nu, modificați înregistrările (în linia de comandă sau cu ajutorul unui script PHP) astfel încât toate cărțile să aibă un id_domeniu și id_autor valid. Vom avea nevoie de toate aceste date în cele ce urmează.

Crearea siteului

Noul site pas cu pas

Puteți să uitați corvoada modificărilor și upload-urilor zilnice: cunoașteți suficient încât să puteți crea un site dinamic și interactiv!

În acest capitol vom face pagină cu pagină secțiunea navigabilă a site-ului nostru, folosind tot ce am învățat până acum. Vom afla pe parcurs mai multe tips and tricks care să ne facă munca și mai ușoară. Ca și până acum, sfatul meu este să scrieți codul prezentat aici și să îl testați voi înșivă, multe concepte pot părea greu de digerat pe hârtie și foarte ușor de înțeles în practică.

Vom face întâi curățenie în Document root (c:\Program files\Apache Group\Apache\htodcs). Ștergeți sau mutați fișierele deja existente acolo pentru a nu vă încurca mai departe. Apoi asigurați-vă că serverul MySQL este pornit și deschideți editorul PHP preferat.

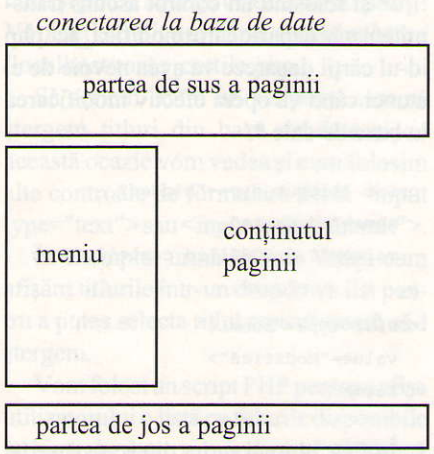
Prima pagină

O primă pagină atractivă este vitală pentru succesul unui site. Menirea ei este să ofere utilizatorului motivele pentru a continua să vadă mai mult și eventual să fie imediat interesat să cumpere ce avem de oferit. Vitrina librăriei noastre virtuale va prezenta utilizatorului atât domeniile de carte disponibile cât și cele mai noi cărți adăugate și precum și cele mai populare. Vom mai avea și o căsuță pentru căutare pentru cei grăbiți sau gata hotărâți pentru ca aceștia să poată găsi rapid ceea ce îi interesează.

Prima pagină este de fapt o combinație din mai multe pagini. Unele elemente sunt prezente pe toate paginile din acest site, ca de exemplu notița de copyright de la sfârșit, instrucțiunile de conectare la baza de date sau prima parte în care specificăm titlul, setul de caractere folosit și stilul CSS.

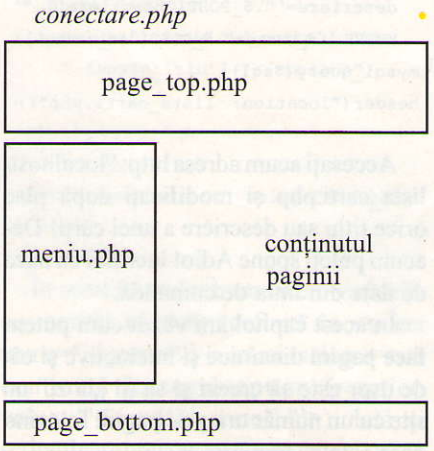
PHP ne oferă funcția include cu care le putem include în cadrul scriptului și să le refolosim în orice alt script în care avem nevoie de ele fără să trebuiască să le scriem din nou.

Structura fiecărei pagini din site-ul nostru, indiferent dacă este prima pagină, pagina de domeniu sau pagina cu detalii despre carte, va arăta astfel:



Această structură o vom folosi pentru toate paginile site-ului și în afară de *conținutul paginii* care diferă, celelalte rămân practic neschimbate. Le vom refolosi, scriind fiecare din ele într-un fișier separat și apoi incluzându-le atunci când avem nevoie de ele.

Astfel, orice pagină din site-ul nostru nu va conține decât elementele relevante care se regăsesc doar în ea: pe homepage cele mai noi cărți, în pagina de domeniu lista cărților din domeniul respectiv și în pagina cu detalii, doar informația despre cartea respectivă. Toate paginile vor avea aceeași structură:



Fișierul conectare.php va conține cele două instrucțiuni, mysql_connect și mysql_select_db pe care le vom folosi pe fiecare pagină a site-ului.

conectare.php

```
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
```

Fisierul page_top.php va conține tagurile clasice cu care se începe un fișier HTML, setul de caractere și titlul paginii, logo-ul și imediat după, deschide tabelul ce urmează să cuprindă meniul și conținutul paginii:

page_top.php

```
<html>
<head>
 <meta  http-equiv="Content-Type"
 content="text/html; charset=iso-8859-2">
 <title>librăria mea</title>
 <style type="text/css">
 body, p, td {font-family: Verdana,
 Arial, sans-serif; font-size: 12px;}
 h1 {font-family: Times New Roman,
 Times, serif; font-size: 18px; font-
 weight: bold; color: #336699; font-
 style:italic;}
 .titlu {font-family: Verdana, Arial,
 sans-serif; font-size: 14px; font-
 weight: bold; color: #0066CC;}
 </style>
</head>
<body bgcolor="#ffffff">
<img src="logo.gif">
<table>
  <tr>
```

În page_bottom.php scriem notița de copyright și închidem toate tagurile deschise.

page_bottom.php

```
  </tr>
</table>
<p align="center"><A href="http://
  www.vogelburda.ro">&copy; Vogel
  Burda Communications</a></p>
</body>
</html>
```

Pagina meniu.php va conține toate domeniile de carte disponibile astfel încât ele să fie accesibile în orice moment de

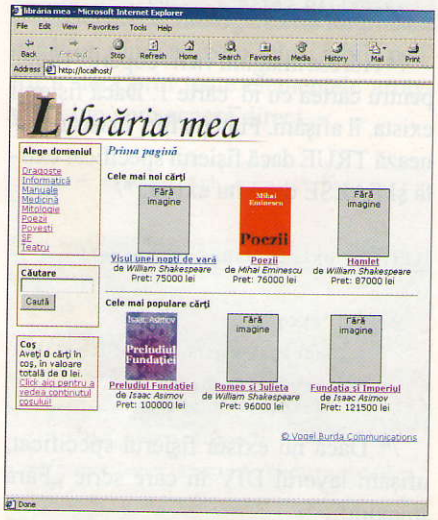
pe orice pagină a site-ului, precum şi caseta de căutare. În acest moment suntem conectaţi la baza de date, deoarece fişierul meniu.php nu îl vom accesa direct ci îl includem **după** ce am inclus fişierul conectare.php. Putem deci să scriem direct scriptul pentru afişarea numelor de domenii, în ordine crescătoare (Aventuri, Biografii, Clasici, etc.).

meniu.php

```
<td valign="top" width="125">
<div style="width:120px; background-
color:#F9F1E7; padding:4px; border:
solid #632415 1px">
<b>Alege domeniul</b><HR size="1">
<?
$sql = "SELECT * FROM domenii ORDER
  BY nume_domeniu ASC";
$resursa = mysql_query($sql);
while($row =
  mysql_fetch_array($resursa))
{
  print '<a href="domeniu.php?
    id_domeniu='.$row['id_domeniu'].
    '">'.$row['nume_domeniu'].
    '</a><br>';
}
?>
</div>
<br>
<div style="width:120px; background-
color:#F9F1E7; padding:4px; border:
solid #632415 1px">
  <form action="cautare.php"
    method="GET">
  <b>Căutare</b><br>
  <INPUT type="text" name="cuvant"
    size="12"><br>
  <INPUT type="submit" value="Caută">
  </form>
</div>
</td>
```

Şi, în final, să creem un fişier index.php care să le cuprindă pe toate. În el vom adăuga conţinut din baza de date astfel încât să avem o primă pagină dinamică. Priviţi pentru un moment cum va arăta pagina noastră după ce vom termina de scris codul. Îmi şi imaginez ce veţi spune: „Are imagini şi până acum nu a spus nimeni nimic despre imagini!". Puteţi transfera fişiere pe server folosind formulare şi PHP însă, cum subiectul depăşeşte tema propusă, vă sugerez să consultaţi manualul PHP de pe CD dacă doriţi să aflaţi mai multe. Mai simplu, putem crea „manual" câte o imagine în format JPEG de 75 de pixeli lăţime şi 100 de pixeli înălţime pentru fiecare copertă a cărţilor din baza de date şi să îi dăm numele 1.jpg pentru coperta cărţii cu id_carte=1, 2.jpg pentru coperta cărţii cu id_carte=2 şi aşa mai departe. Toate aceste imagini le vom pune într-un director numit coperte în document root.

Astfel, când afişăm informaţia despre cartea cu id_carte=3 putem să îi afişăm şi coperta folosind <img src="coperte/3.jpg">. Pentru mai multă siguranţă, înainte de a afişa imaginea, scriptul nostru se va uita în directorul coperte şi doar dacă va găsi imaginea, o va afişa.



Prima pagină conţine cele mai importante elemente.

index.php

```
<?
include("conectare.php");
include("page_top.php");
include("meniu.php");
?>
<td valign="top">
<h1>Prima pagină</h1>
<b>Cele mai noi cărţi</b>
  <table cellpadding="5">
    <tr>
<?
```

/* În următoarea interogare selectăm informaţiile despre cele mai noi trei cărţi şi le afişăm pe fiecare într-o celula de tabel: */

```
$sql = "SELECT id_carte, titlu,
  nume_autor, pret FROM carti, autori
  WHERE carti.id_autor=autori.id_autor
  ORDER BY data DESC LIMIT 0,3";
$resursa = mysql_query($sql);
while($row =
  mysql_fetch_array($resursa))
{
  /* deschidem celula tabelului
  HTML */
  print '<td align="center">';
```

/* Să punem şi imaginea copertei.

Este posibil ca nu pentru toate cărţile să avem o imagine, aşa că vom face după cum urmează: dacă avem imagine pentru copertă, o afişăm. Dacă nu, afişăm un

Funcţia include

Mai uşor decât Copy&Paste

Cu ajutorul funcţiei include putem refolosi bucăţi de cod fără să fie nevoie să le scriem ori de câte ori avem nevoie de ele, indiferent dacă este cod PHP precum instrucţiunile de conectare la baza de date sau HTML precum notiţa de copyright de pe fiecare pagină a site-ului.

Trebuie doar să scriem părţile refolosibile în fişiere separate pe care să le accesăm atunci când avem nevoie de ele. Iată cum includem trei fişiere în altul:

top.html

```
<html>
<body bgcolor="#FFFFCC">
<h1>titlul paginii</h1>
```

copyright.html

```
<p><a href="http://
www.vogelburda.ro">&copy Vogel Burda
Communications</a></p>
```

variabile.php

```
<?
print "un text conţinut în fişierul
variabile.php";
$variabila = "şi o variabilă din
fişierul variabile.php";
?>
```

Şi iată şi fişierul care le foloseşte pe toate acestea:

test1.php

```
<?
include("top.html");
?>
<p>Un text oarecare!</p>
<?
include("variabile.php");
print $variabila1;
include("copyright.html");
?>
```

Observăm că fişierele care conţin cod PHP trebuie să conţină tagurile <? şi ?>, aceasta pentru că parserul iese din modul PHP la începutul fişierului inclus şi trece în mod HTML, reluând modul PHP atunci când termină de evaluat fişierul respectiv. Variabilele, funcţiile şi resursele din fişierele incluse sunt disponibile în cadrul scriptului după ce au fost incluse. Dacă setarea URL_fopen_wrappers este activată în php.ini puteţi include chiar şi fişiere aflate pe alte servere: include("http://www.chip.ro").

layer DIV în care scriem simplu „Fără imagine". Vom compune întâi adresa imaginii, ştiind ca aceasta se află în directorul coperte, are acelaşi nume ca id_carte şi extensia jpg: */

```
$adresaImagine = "coperte"
  .$row['id_carte'].".jpg";
```

/* Adresa imaginii va fi "coperte/1.jpg" pentru cartea cu id_carte 1. Dacă fişierul exista, îl afişăm. Funcţia file_exists returnează TRUE dacă fişierul specificat există şi FALSE dacă nu există. */

```
if(file_exists($adresaImagine))
{
  print '<img src=
    "'.$adresaImagine.'" width="75"
    height="100"><br>';
}
```

/* Dacă nu există fişierul specificat, afişăm layerul DIV în care scrie „Fără imagine": */

```
else
{
print '<div style="width:75px;
  height:100px; border: 1px black
solid; background-color:#cccccc">
Fără imagine</div>';
}
```

/* Să continuăm cu afişarea restului de informaţii, titlul, numele autorului, preţul: */

```
print '<b><a href="carte.php?
  id_carte='.$row['id_carte'].'">'
  .$row['titlu'].'</a></b><br> de
  <i>'.$row['nume_autor'].'</i><br>
  Pret: '.$row['pret'].' lei
```

/* Închidem celula <td> cu care am început structura while. While va afişa toate cele trei cărţi selectate în interogarea SQL şi pentru fiecare din ele va repeta paşii de mai sus. */

```
  </td>';
}
```

/* Am terminat cu cele mai noi cărţi. Închidem tabelul, tragem o linie de separare după care vom scrie acelaşi cod pentru cele mai populare cărţi, modificând doar interogarea SQL. */

```
?>
    </tr>
  </table>
<hr>
<b>Cele mai populare carti</b>
<table cellpadding="5">
  <tr>
  <?
```

/* Care sunt cele mai populare cărţi? Cele mai vândute, desigur. Aflăm care sunt cele mai vândute cărţi, consultând tabelul vânzări, cu interogarea următoare. Dacă nu aveţi nici un fel de date în tabelul vânzări, interogarea va returna 0 rânduri (nici o carte vândută) şi nu va apărea nimic pe prima pagină în secţiunea „Cele mai populare cărţi". Introduceţi câteva date în acest tabel. */

```
$sqlVanzari = "SELECT id_carte,
  sum(nr_buc) AS bucatiVandute FROM
  vanzari GROUP BY id_carte ORDER BY
  bucatiVandute DESC LIMIT 0,3";
```

/* Notă explicativă privind această interogare: din tabelul vânzări ne interesează două coloane: id_carte şi nr_buc (cărţi vândute). Dacă grupăm id-urile, putem afla numărul de total de bucăţi vândute din fiecare carte folosind funcţia MySQL sum(nr_buc). Id-urile le ordonăm apoi descrescător în funcţie de numărul total de bucăţi vândute din fiecare (ORDER BY bucatiVandute). bucatiVandute este de fapt un alias pentru coloana creată ad-hoc, sum(nr_buc).

Definirea unui alias se poate face folosind AS ca în SELECT sum(nr_buc) AS bucatiVandute. Am definit această coloană creată instant ca alias deoarece nu am fi putut face ordonarea direct ORDER BY sum(bucatiVandute). Puteţi rula această interogare în linia de comandă pentru a vedea rezultatul aşa cum îl oferă MySQL. Să continuăm cu scriptul PHP: */

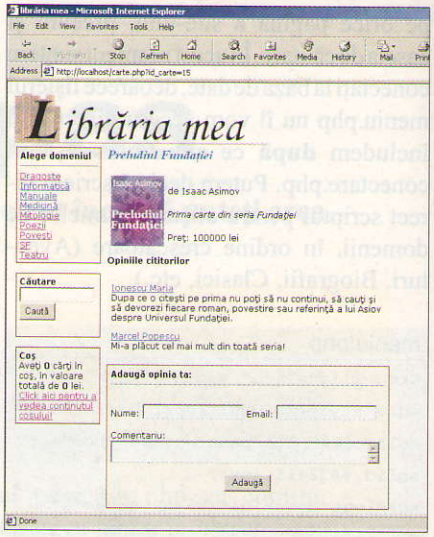
```
$resursaVanzari=mysql_query($sqlVanzari);
```

/* Valorile din acest query sunt trei id_carte din tabelul vânzări care corespund celor mai trei vândute cărţi şi numărul total de bucaţi vândute din fiecare. Vom folosi aceste id-uri pentru a interoga, cu fiecare din ele, baza de date şi a afla titlul, autorul şi preţul fiecăreia din ele:*/

```
while($rowVanzari =
  mysql_fetch_array($resursaVanzari))
{
  $sqlCarte = "SELECT titlu,
    nume_autor, pret FROM carti,
    autori WHERE carti.id_autor=
    autori.id_autor AND id_carte=".
    $rowVanzari['id_carte'];
  $resursaCarte=mysql_query($sqlCarte);
```

/* Acum avem toate datele care ne interesează: id_carte (din interogarea $sqlVanzari), titlul, numele autorului şi numărul de bucăţi vândute.

Să le afişăm: */



Comentariile utilizatorilor la această carte.

```
while($rowCarte =
  mysql_fetch_array($resursaCarte))
{
  print '<td align="center">';
  $adresaImagine = "coperte/
    ".$rowVanzari['id_carte'].".jpg";
  if(file_exists($adresaImagine))
  {
    print '<img
      src="'.$adresaImagine.'"
      width="75" height="100"><br>';
  }
  else
  {
    print '<div style="width:75px;
      height:100px; border: 1px black
      solid; background-color:#cccccc">
      Fără imagine</div>';
  }
  print '<b><a
    href="carte.php?id_carte='.
    $rowVanzari['id_carte'].'">'.
    $rowCarte['titlu'].'</a></b><br> de
    <i>'.$rowCarte['nume_autor'].'</i>
    <br>Pret: '.$rowCarte['pret'].' lei
    </td>';
  }
}
?>
    </tr>
  </table>
 </td>
</tr>
</table>
<?
include("page_bottom.php");
?>
```

Acesta a fost index.php, prima pagină a site-ului nostru. Întotdeauna informaţiile de pe ea vor fi la zi. Puteţi adăuga o

carte nouă în baza de date pentru a vedea că apare imediat apoi şi pe prima pagină (nu uitaţi să menţionaţi şi data deoarece acesta este criteriul de ordonare!).

Domenii

A doua pagină pe care o vom scrie este cea care se ocupă de domenii. Am şi scris deja, în capitolul anterior, un mic script care să ne afişeze rapid titlurile cărţilor dintr-un domeniu ales. Acelaşi lucru vom face în continuare, ceva mai elaborat. În pagina domeniu.php vom afişa cărţile ce aparţin domeniului selectat, pozele coperţelor, numele autorilor şi preţul. În acest script doar interogarea SQL este mai complicată deoarece „leagă" trei tabele (cărţi, domenii, autori) dar restul ar trebui să vă pară deja floare la ureche! Iată şi un mic tip înainte de a începe. Atunci când doriţi să afişaţi valoarea unei variabile în cadrul codului HTML, în loc să scrieţi:

```
cod HTML
<?
print $numeVariabila;
?>
cod HTML
```

puteţi apela la mult mai simplul:

```
cod HTML <?=$numeVariabila?> cod HTML
```

Să trecem la treabă şi să folosim în următorul script această nouă facilitate:

domeniu.php

```
<?
include("conectare.php");
include("page_top.php");
include("meniu.php");
$id_domeniu = $_GET['id_domeniu'];
$sqlNumeDomeniu = "SELECT
  nume_domeniu FROM domenii WHERE
  id_domeniu=".$id_domeniu;
$resursaNumeDomeniu =
  mysql_query($sqlNumeDomeniu);
$numeDomeniu = mysql_result
  ($resursaNumeDomeniu,0,"nume_domeniu");
?>
<td valign="top">
<h1>Domeniu: <?=$numeDomeniu?></h1>
<b>Cărţi în domeniul
<u><i><?=$numeDomeniu?></i></u>:</b>
<table cellpadding="5">
<?
$sql = "SELECT id_carte, titlu,
  descriere, pret, nume_autor FROM
  carti, autori, domenii WHERE
  carti.id_domeniu=domenii.id_domeniu
  AND carti.id_autor=autori.id_autor
  AND domenii.id_domeniu=".$id_domeniu;
$resursa = mysql_query($sql);
while($row=mysql_fetch_array($resursa))
{
 ?>
 <tr>
  <td align="center">
  <?
  $adresaImagine = "coperte"
   .$row['id_carte'].".jpg";
  if(file_exists($adresaImagine))
  {
   print '<img src=
    "'.$adresaImagine.'" width="75"
    height="100"><br>';
  }
  else
  {
   print '<div style="width:75px;
    height:100px; border: 1px black
    solid; background-color:#cccccc">
    Fără imagine</div>';
  }
  ?>
  </td>
  <td>
   <b><A href="carte.php?
   id_carte=<?=$row['id_carte']?>">
   <?=$row['titlu']?></a></b><br>
   <i>de <?=$row['nume_autor']?></i>
   <br>Preţ: <?=$row['pret']?> lei
  </td>
 </tr>
 <?
}
?>
  </table>
</td>
<?
include("page_bottom.php");
?>
```

Mai avem de făcut o pagină în care să prezentăm detaliile cărţii, să îi dăm posibilitatea utilizatorului să adauge cartea în coş, să citească părerile altora despre cartea respectivă sau să adauge propria sa opinie. În acest script vom face referire la două fişiere: comentarii.php (care preia un comentariu trimis prin formular, îl prelucrează şi îl introduce în baza de date) şi coş.php, pagina cu ajutorul căreia vom adăuga sau scoate cărţi din coş şi îi vom putea vizualiza conţinutul. Dar să scriem întâi codul pentru carte.php:

carte.php

```
<?
include("conectare.php");
include("page_top.php");
include("meniu.php");
$id_carte = $_GET['id_carte'];
$sql = "SELECT titlu, nume_autor,
  descriere, pret FROM carti, autori
  WHERE id_carte=".$id_carte." AND
  carti.id_autor=autori.id_autor";
$resursa = mysql_query($sql);
```

// Deoarece interogarea nu returnează decât un rând, nu vom folosi while pentru a itera prin toate elementele array-ului ci le vom accesa direct.

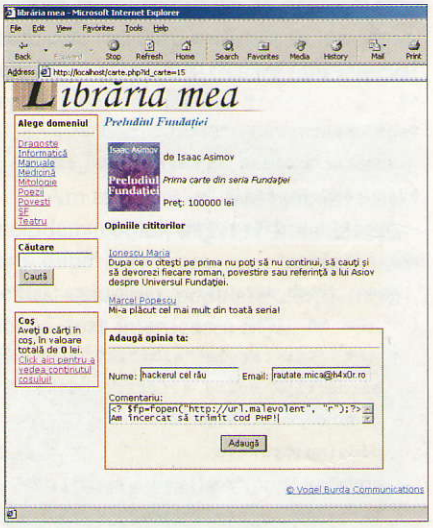
```
$row = mysql_fetch_array($resursa);
?>
<td valign="top">
 <table>
  <tr>
   <td valign="top">
    <?
    $adresaImagine = "coperte/"
     .$id_carte.".jpg";
    /*Dacă avem imagine pentru co-
    pertă, o afişăm, iar dacă nu
    avem, nu afişăm nimic:*/
    if(file_exists($adresaImagine))
    {
     print '<img src=
      "'.$adresaImagine.'" width="75"
      height="100" hspace="10"><br>';
    }
    ?>
   </td>
   <td valign="top">
    <h1><?=$row['titlu']?></h1>
    <i>de <b><?=$row['nume_autor']?>
     </b></i>
    <p><i><?=$row['descriere']?>
     </i></p>
    <p>Pret: <?=$row['pret']?> lei</p>
   </td>
  </tr>
 </table>
<p><b>Opiniile cititorilor</b></p>
<?
$sqlComentarii = "SELECT * FROM
 comentarii WHERE id_carte=".$id_carte;
$resursaComentarii =
  mysql_query($sqlComentarii);
while($row =
  mysql_fetch_array($resursaComentarii))
{
  print '<div style="width:400px;
   border:1px solid #ffffff; back-
   ground-color:#F9F1E7;
   padding:5px"><a
   href="mailto:'.$row['adresa_email'].'">'
   .$row['nume_utilizator'].'</a>
   <br>'.$row['comentariu'].'
   </div> ';
}
```

```
?>
<br>
<div style="width:400px; border:1px
  solid #632415; background-
  color:#F9F1E7; padding:5px;">
<b>Adaugă opinia ta:</b>
<hr size="1">
<form action="adauga_comentariu.php"
  method="POST">
  Nume: <input type="text"
    name="nume_utilizator">
  Email: <input type="test"
    name="adresa_email"><br><br>
  Comentariu:<br>
    <textarea name="comentariu"
    cols="45"></textarea><br><br>
  <input type="hidden"
    name="id_carte"
    value="<?=$id_carte?>">
  <center><input type="submit"
    value="Adaugă"></center>
</form>
</div>
</td>
<?
include("page_bottom.php");
?>
```

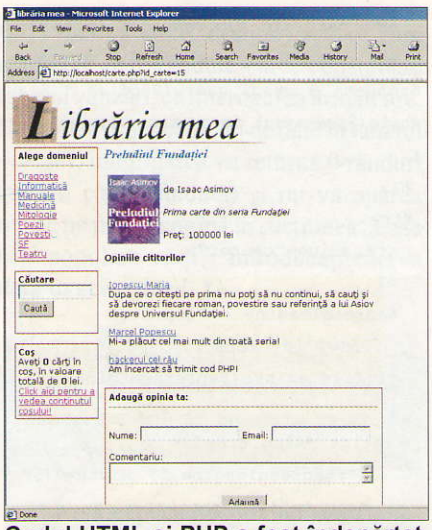
În pagina fiecărei cărți avem un formular cu ajutorul căruia utilizatorii să-şi poată împărtăşi impresiile precum şi lista impresiilor deja adăugate. Iată fişierul adauga_comentariu.php care prelucrează informaţia trimisă prin formular şi o introduce în baza de date:

adauga_comentariu.php

/* Verifică dacă toate câmpurile formularului de comentarii au fost completate şi dacă oricare din ele a fost omis, afişează mesajul „Trebuie să completezi toate câmpurile!" şi întrerupe execuţia



Putem preveni multe probleme dacă folosim strip_tags pentru a elimina tagurile din inputul utilizatorilor.



Codul HTML şi PHP a fost îndepărtat. Comentariul apare ca text simplu!

scriptului */

```
<?
if($_POST['nume_utilizator'] == ""
  || $_POST['adresa_email'] == ""
  || $_POST['comentariu'] == "")
{
   print "Trebuie să completezi
     toate câmpurile!";
   exit;
}
```

/* Dacă execuţia scriptului a ajuns până aici (adică a trecut cu succes de condiţia de mai sus) înseamnă că toate câmpurile au fost completate.

Aşadar, se conectează la baza de date, prelucrează informaţiile transmise din formular şi le introduce în baza de date:*/

```
include("conectare.php");
```

/* Folosim din motive de securitate funcţia strip_tags pentru a elimina tagurile HTML şi PHP din toate stringurile trimise de utilizator.

E bine să folosim strip_tags pentru a „curăţa" inputul utilizatorilor de orice cod potential răuvoitor.*/

```
$numeFaraTags =
  strip_tags($_POST['nume_utilizator']);
$emailFaraTags =
  strip_tags($_POST['adresa_email']);
$comentariuFaraTags =
  strip_tags($_POST['comentariu']);
$sql = "INSERT INTO comentarii
  (id_carte, nume_utilizator,
  adresa_email, comentariu)
  VALUES(".$_POST['id_carte'].",
  '".$numeFaraTags."','".$emailFaraTags."',
  '".$comentariuFaraTags."')";
mysql_query($sql);
```

/* Apoi redirecţionăm utilizatorul către pagina cărţii la care a adăugat un comentariu:*/

```
$inapoi = "carte.php?id_carte=".
  $_POST['id_carte'];
header("location: $inapoi");
?>
```

Coşul de cumpărături

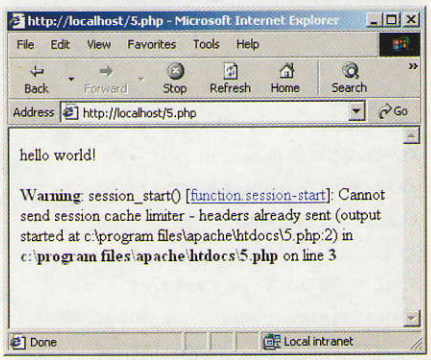
Nu am uitat că menirea principală a acestui site nu este doar de a prezenta cărţi şi a oferi posibilitatea ca utilizatorii să-şi împărtăşească impresiile despre ea ci şi să cumpere! Vom face o singură pagină pentru coşul de cumpărături, pagină cu ajutorul căreia vom putea adăuga, modifica sau scoate cărţi din coş şi asta în mai puţin de 100 de linii de cod. Dar înainte de a începe ne întrebăm, pe bună dreptate: dacă vizitatorul adaugă o carte în coş dar apoi doreşte să cumpere altă carte, unde se păstrează informaţiile despre prima carte vândută? Nu se pierd odată ce a ieşit din pagina cos.php?

Am putea oare să le păstrăm în baza de date? Am putea dar pentru asta ar trebui să corelăm informaţia din baza de date cu utilizatorul şi ar fi destul de complicat.

Am putea să trimitem informaţiile despre cartea cumpărată mai departe prin URL? Şi aşa ar fi prea complicat. Din fericire, PHP ne oferă o metodă foarte uşoară şi la îndemână de a păstra informaţiile, pe tot timpul cât este vizitat site-ul: sesiunile.

Sesiuni

Sesiunile sunt mijlocul prin care putem păstra o serie de informaţii în memorie. Astfel, cumpărătorul nostru poate adăuga o carte în coş şi apoi să continue să viziteze site-ul. Informaţiile despre cartea sau cărţile deja adăugate în coş se vor



Sesiunea nu poate fi pornită după ce am trimis deja un output către browser.

păstra în memorie şi noi le putem accesa doar atunci când este nevoie. Sesiunile expira după 3 ore (puteţi lăsa browserul deschis peste noapte şi dimineaţa să reîncărcaţi o pagină care afişează nişte variabile de sesiune pentru a vedea că nu sunt acolo) sau după ce toate instanţele browserului sunt închise. Într-o sesiune datele pot fi salvate într-o variabilă de tip array, numită $_SESSION. Înainte de a folosi această variabilă pentru a stoca informaţii trebuie să apelăm funcţia predefinită session_start(). session_start porneşte o sesiune dacă nu există niciuna sau o reiniţializează/continuă pe cea existentă, dacă aceasta a fost deja pornită.

Informaţiile din sesiune, spre deosebire de cookie-uri, sunt stocate pe server directorul pentru fişiere temporare. Pe unele sisteme Windows, dacă directorul temporar nu este setat cum trebuie, va trebui să deschideţi php.ini şi să setaţi chiar voi directorul temporar unde să fie păstrate sesiunile. Pentru a vedea dacă configuraţia dvs. este corectă, creaţi un nou fişier sesiune.php în care scrieţi:

```
<?
session_start();
?>
```

Rulaţi-l în browser. Dacă primiţi un mesaj de eroare (Warning!), va trebui să setaţi în php.ini directorul temporar. Modificaţi linia

```
session.save_path=/tmp
```

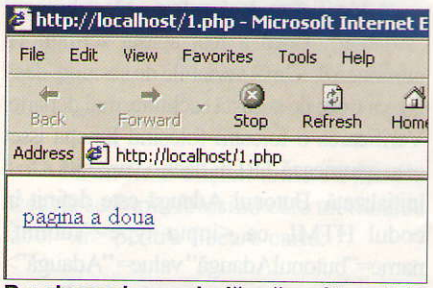
în

```
session.save_path=c:\temp\
```

Poate fi necesar să reporniţi serviciul Apache pentru ca modificările să fie aplicate. Accesaţi acum pagina. Dacă este goală şi nu aţi primit nici un mesaj de eroare, configuraţia dvs. este corectă.

De ajuns cu „limba de lemn", să vedem sesiunile la lucru.

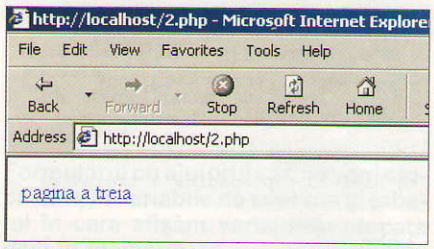
În pagina 1.php iniţializăm sesiunea şi stocăm două valori:



Pornim sesiunea şi adăugăm câteva date în ea, nu afişăm nimic utilizatorului

1.php

```
<?
session_start();
$_SESSION['titlu'] = "Dune";
$_SESSION['autor'] = "Frank Herbert";
?>
<a href="2.php">pagina a doua</a>
```

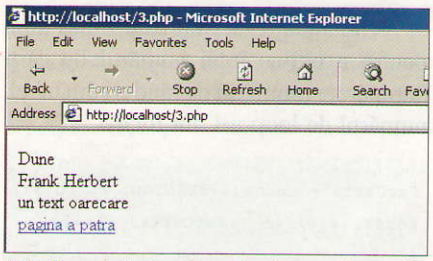


Reiniţializăm sesiunea pornită în pagina anterioară şi mai adăugăm date în ea.

În pagina 2.php continuăm sesiunea folosind session_start după care mai stocăm o variabilă:

2.php

```
<?
session_start();
$_SESSION['altaVariabila'] = "un text
  oarecare";
?>
<a href="3.php">pagina a treia</a>
```



Afişăm datele stocate în sesiune în paginile anterioare.

În pagina 3.php folosim din nou session_start după care afişăm toate datele stocate până acum în această sesiune:

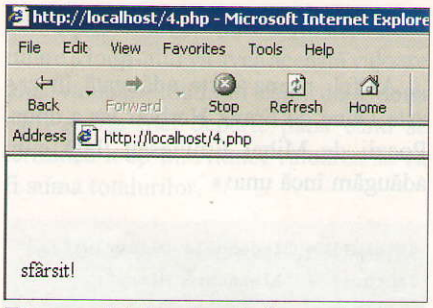
3.php

```
<?
session_start();
print $_SESSION['titlu']."<br>";
print $_SESSION['autor']."<br>";
print $_SESSION['altaVariabila']."<br>";
?>
<a href="4.php">pagina a patra</a>
```

Toate variabilele salvate în această sesiune au fost afişate! Iată şi 4.php:

4.php

```
<?
print $_SESSION['titlu']."<br>";
```



Eroare: datele stocate în sesiune nu sunt disponibile deoarece nu am reiniţializat sesiunea.

```
print $_SESSION['autor']."<br>";
print $_SESSION['altaVariabila']."<br>";
?>
sfârsit!
```

De ce nu vedem valorile şi pe a patra pagină? Pentru că nu am continuat sesiunea cu session_start. Întotdeauna trebuie să reinitializăm sesiunea folosind session_start inainte de a adăuga sau manipula valori în sesiune! Pe lângă aceasta, mai există o restricţie: session_start trebuie folosit intotdeauna înainte de a trimite date către browser altfel primim un mesaj de eroare:

5.php

```
<?
print "hello world!<br>";
session_start();
?>
```

Această regulă este valabilă chiar şi pentru tag-ul <html> care nu produce un efect vizibil în browser, exemplul următor va da aceeaşi eroare:

6.php

```
<html>
<?
session_start();
?>
```

Ţineţi minte, întotdeauna rulaţi session_start înainte de a trimite orice output către browser!

Sesiunile ne pot ajuta să păstrăm in memorie informaţiile despre cărţile deja cumpărate de către un vizitator. Cum salvăm totuşi mai mult decât o carte? Folosind variabilele de sesiune pentru a stoca array-uri. Iată, de exemplu, cum am putea stoca variabilele corespunzătoare fiecărei cărţi, ca un array multidimensional:

```
$titlu[1] = "Dune";
$autor[1] = "Frank Herbert";
$titlu[2] = "Poezii";
```

```
$autor[2] = "Mihai Eminescu";
```

Astfel, prima carte adăugată în coş este Dune de Frank Herbert iar a doua, Poezii de Mihai Eminescu. Iată cum adăugăm încă una:

```
$titlu[] = "Legendele Olimpului";
$autor[] = "Alexandru Mitru";
```

N-am specificat nici un număr deoarece folosind [] datele sunt adăugate automat la sfârşitul array-ului şi astfel vom avea valoarea „Legendele Olimpului" pentru $titlu[3].

Operatorul [] pentru adăugarea de valori într-un array ne va folosi pentru a nu ţine tot timpul minte ce număr se află între parantezele []. Putem să adăugăm valori într-un array începând de la 0, doar folosind [], nespecificând nici un număr. Iată spre exemplu cum am putea accesa valorile array-ului specificând valoarea indexului numeric:

7.php

```
$titlu[] = "Dune";
$titlu[] = "Poezii";
$titlu[] = "Legendele Olimpului";
print $titlu[0]; // va afişa Dune
print $titlu[1]; // va afişa Poezii
print $titlu[2]; // va afişa Legendele
  Olimpului
```

Şi iată cum accesăm toate valorile array-ului, folosind for:

```
$titlu[] = "Dune";
$titlu[] = "Poezii";
$titlu[] = "Legendele Olimpului";
$nrElementeInArray = count($titlu);
for($i = 0; $i < $nrElementeInArray;
  $i++)
{
  print $titlu[$i]."<br>";
}
```

Astfel, pentru $i = 0, va fi afişată valoarea lui $titlu[0], pentru $i = 1 valoarea lui $titlu[1] şi aşa mai departe până când când se ajunge la limita reprezentată de numărul de elemente din array. La fel vom folosi array-uri pentru a salva în variabile de sesiune informaţiile despre cărţile cumpărate. Să facem întâi un mic exerciţiu. Să trecem prin 3 fişiere, presupunând că în fiecare se cumpără câte o carte:

1.php

```
<?
session_start();
$_SESSION['titlu'][] = "Dune";
$_SESSION['autor'][] = "Frank Herbert";
$_SESSION['pret'][] = 100000;
$_SESSION['nr_buc'][] = 1;
?>
<A href="2.php">pagina a doua</a>
```

2.php

```
<?
session_start();
$_SESSION['titlu'][] = "Poezii";
$_SESSION['autor'][] = "Mihai Eminescu";
$_SESSION['pret'][] = 100000;
$_SESSION['nr_buc'][] = 1;
?>
<A href="3.php">pagina a treia</a>
```

3.php

```
<?
session_start();
$_SESSION['titlu'][] = "Manual de
  geografie";
$_SESSION['autor'][] = "Popescu Ion";
$_SESSION['pret'][] = 50000;
$_SESSION['nr_buc'][] = 30;
```

/* avem până acum trei cărţi cumpărate, să şi trecem la afişarea lor folosind for, folosind câte un rând de tabel <tr> pentru fiecare carte: */

```
print '<table border="1">';
```

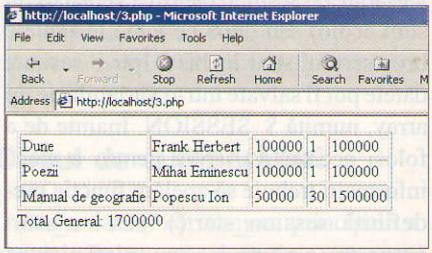
/* ne ajunge să ştim câte titluri sunt pentru a obţine întâi numărul de cărţi cumpărate (numărul de rânduri din tabel, numărul de loop-uri din for)*/

```
$nrCarti = count($_SESSION['titlu']);
for($i = 0; $i < $nrCarti; $i++)
{
  print '<tr>';
  print '<td>'.
    $_SESSION['titlu'][$i].'</td>';
  print '<td>'.
    $_SESSION['autor'][$i].'</td>';
  print '<td>'.
    $_SESSION['pret'][$i].'</td>';
  print '<td>'.
    $_SESSION['nr_buc'][$i].'</td>';
```

/* calculăm şi cât este totalul pentru această carte, pentru Poezii va fi 1 buc x 100000 lei = 100000 lei în timp ce pentru Manualul de geografie va fi 30 buc x 50000 lei = 1500000 lei*/

```
$total = $_SESSION['nr_buc'][$i] *
    $_SESSION['pret'][$i];
print '<td>'.$total.'</td>';
print '</tr>';
```

/* înainte de a încheia loop-ul, vom adăuga această valoare la totalul general. Variabila $totalGeneral va fi iniţializată la prima rulare a loop-ului şi va avea valoarea totalului primei cărţi. La a doua



Datele din array, afişate sub formă de tabel.

rulare a loop-ului va avea această valoare plus valoarea totalului celei de-a doua cărţi şi aşa mai departe până când se termină loop-ul. Atunci valoarea sa va fi suma totalurilor. */

```
  $totalGeneral = $totalGeneral +
    $total;
}
print '</table>';
print 'Total General: '.$totalGeneral;
?>
```

Să vedem acum cum am putea folosi un singur fişier pentru a adăuga cărţi în tabel. Vom profita de faptul că orice cod PHP se execută pe server şi abia apoi output-ul său este trimis către browser. Să facem un fişier test.php în care vom avea un formular al cărui action să fie chiar test.php. Înainte de a afişa formularul, însă, vom adăuga în variabile de sesiune datele trimise chiar prin formular. Dacă aceasta este prima accesare a scriptului, nici un fel de date nu vor fi fost trimise, şi vom trece peste codul de adăugare în variabile de sesiune. Dacă apoi completăm câmpurile formularului şi apăsăm butonul „Adaugă", scriptul de la începutul paginii va recunoaşte că sunt date trimise, le va salva în variabile de sesiune şi apoi le va afişa într-un tabel, după formular.

test.php

```
<?
  // Pornim sesiunea
session_start();
```

/* Verificăm dacă a fost apăsat butonul „Adaugă". Dacă a fost apăsat scriptul va introduce datele în variabile de sesiune, dacă nu, va trece de această secţiune, mai departe. Verificarea o folosim folosind funcţia isset care returnează TRUE dacă o variabilă a fost iniţializată. Butonul Adaugă este definit în codul HTML ca <input type="submit" name="butonulAdaugă" value="Adaugă"> iar variabila trimisă către scriptul PHP va fi $_POST['butonulAdauga'] ="Adaugă". Să

verificăm, deci, dacă variabila a fost trimisă (dacă a fost apăsat butonul)*/

```
if(isset($_POST['butonulAdauga']))
{
```

/* Dacă butonul a fost apăsat, se execută acest cod. Dacă nu, scriptul ignoră codul dintre parantezele {} lui if şi trece mai departe */

```
$_SESSION['titlu'][] = $_POST['titlu'];
$_SESSION['autor'][] = $_POST['autor'];
$_SESSION['nr_buc'][] = $_POST['nr_buc'];
$_SESSION['pret'][] = $_POST['pret'];
```

/* Am adăugat toate datele trimise în POST în variabile de sesiune */

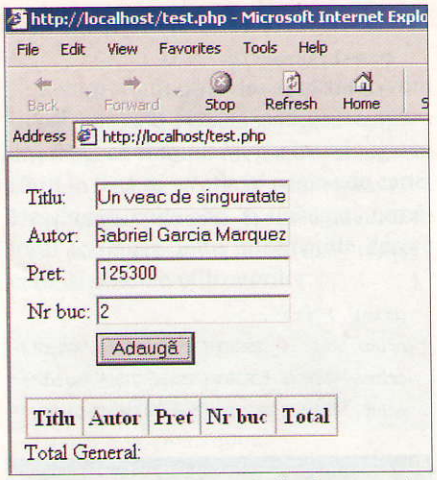
```
}
```

/* Să afişăm acum formularul a cărui acţiune va fi acelaşi fişier:*/

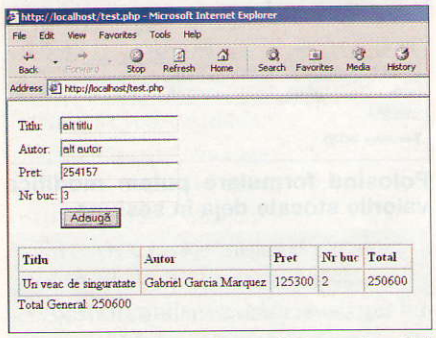
```
?>
<form action="test.php"
  method="post">
 <table>
  <tr>
   <td>Titlu:</td>
   <td><input type="text"
     name="titlu"></td>
  </tr>
  <tr>
   <td>Autor:</td>
   <td><input type="text"
     name="autor"></td>
  </tr>
  <tr>
   <td>Pret:</td>
   <td><input type="text"
     name="pret"></td>
  </tr>
  <tr>
   <td>Nr buc:</td>
   <td><input type="text"
     name="nr_buc"></td>
  </tr>
  <tr>
   <td></td>
   <td><input type="submit"
     name="butonulAdauga"
     value="Adaugă"></td>
  </tr>
 </table>
</form>
<?
```

/* Iar în final afişăm toate cărţile din această sesiune, folosind câte un rând de tabel <tr> pentru fiecare carte: */

```
print '<table border="1"
  cellspacing="0" cellpadding="4">';
```



Formularul cu ajutorul căruia vom stoca date în variabile de sesiune şi tabelul în care afişăm variabilele stocate deja în memorie.



Datele trimise cu ajutorul formularului au fost adăugate în variabile de sesiune iar apoi afişate în tabel.

```
$nrCarti=count($_SESSION['titlu']);
for($i = 0; $i < $nrCarti; $i++)
{
  print '<tr>';
  print '<td>'.$_SESSION['titlu'][$i]
    .'</td>';
  print <td>'.$_SESSION['autor'][$i]
    .'</td>';
  print '<td>'.$_SESSION['pret'][$i]
    .'</td>';
  print <td>'.$_SESSION['nr_buc'][$i]
    .'</td>';
```

/* calculăm şi cât este totalul pentru această carte, pentru Poezii va fi 1 buc x 100000 lei = 100000 lei în timp ce pentru Manualul de geografie va fi 30 buc x 50000 lei = 1500000 lei*/

```
$total = $_SESSION['nr_buc'][$i] *
   $_SESSION['pret'][$i];
print '<td>'.$total.'</td>';
print '</tr>';
```

/* înainte de a încheia loop-ul, vom adăuga această valoare la totalul general. Variabila $totalGeneral va fi iniţializată la prima rulare a loop-ului şi va avea valoarea totalului primei cărţi. La a doua rulare a loop-ului va avea această valoare plus valoarea totalului celei de-a doua cărţi şi aşa mai departe până când se termină loop-ul. Atunci valoarea sa va fi suma totalurilor. */

```
  $totalGeneral = $totalGeneral + $total;
}
print '</table>';
print 'Total General: '.$totalGeneral;
?>
```

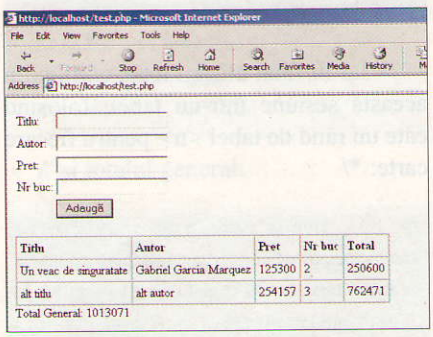
Cum am putea folosi acelaşi fişier şi pentru a modifica sau a şterge cărţi din listă? Putem include tabelul cu lista cărţilor într-un formular şi să folosim elemente de tip <input type="text"> pentru valorile care specifică numărul de bucăţi pentru fiecare carte în parte. Ne vom folosi de faptul că elementele array-ului $_SESSION['pret'] sunt ordonate crescător, începând de la 0 pentru a specifica în elementele input şi indexul numeric al array-ului care ne va fi folositor pentru a opera modificările.

```
<?
  // Pornim sesiunea
session_start();
```

/* Dacă a fost apăsat butonul „Adaugă" executăm acest cod pentru a adăuga cărţi în listă, iar dacă nu, acest cod este ignorat: */

```
if(isset($_POST['butonulAdauga']))
{
  $_SESSION['titlu'][] = $_POST['titlu'];
  $_SESSION['autor'][] = $_POST['autor'];
  $_SESSION['nr_buc'][] = $_POST['nr_buc'];
  $_SESSION['pret'][] = $_POST['pret'];
}
```

/* Dacă a fost apăsat butonul „Modifică" executăm acest cod pentru a modifica valori, iar dacă nu, codul următor este ignora. Dacă a fost apăsat, înseamă că s-au trimis valorile pentru câmpurile nr_buc. Folosim indexul numeric speci-



Tabelul afişează toate variabilele de sesiune.

ficat atunci când vom fi creat câmpurile <input> ale formularului.*/

```
if(isset($_POST['butonulModifica']))
{
  for($i = 0; $i <
    count($_SESSION['titlu']); $i++)
  {
    $_SESSION['nr_buc'][$i] =
      $_POST['nr_buc'][$i];
  }
}
```

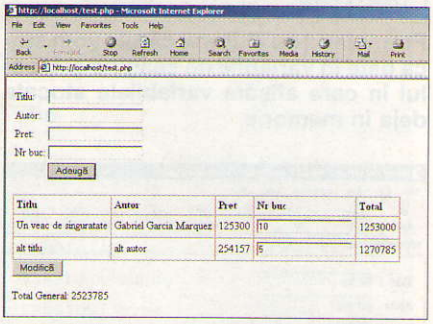
/* Am folosit for pentru a trece prin toate elementele array-ului $_SESSION ['nr_buc'] deoarece este cel mai simplu: toate valorile se trimit prin formular indiferent dacă sunt modificate sau nu. Noi le actualizăm pe toate, cu valorile modificate sau nu.*/

/* Să afişăm acum formularul a cărui acţiune va fi acelaşi fişier:*/

```
?>
<form action="test.php" method="post">
 <table>
  <tr>
    <td>Titlu:</td>
    <td><input type="text"
    name="titlu"></td></tr>
  <tr>
    <td>Autor:</td>
    <td><input type="text"
    name="autor"></td></tr>
  <tr>
    <td>Pret:</td>
    <td><input type="text"
    name="pret"></td></tr>
  <tr>
    <td>Nr buc:</td>
    <td><input type="text"
    name="nr_buc"></td></tr>
  <tr>
    <td></td>
    <td><input type="submit"
    name="butonulAdauga"
    value="Adaugă"></td></tr>
 </table>
</form>
<?
```

/* Iar în final afişăm toate cărţile din această sesiune într-un tabel, folosind câte un rând de tabel <tr> pentru fiecare carte: */

```
?>
<form action="test.php" method="post">
<table border="1" cellspacing="0"
  cellpadding="4">
<?
print '<tr>
  <td><b>Titlu</b></td>
  <td><b>Autor</b></td>
  <td><b>Pret</b></td>
  <td><b>Nr  buc</b></td>
  <td><b>Total</b></td>
  </tr>';
$nrCarti=count($_SESSION['titlu']);
for($i = 0; $i < $nrCarti; $i++)
{
 print '<tr>';
 print '<td>'.$_SESSION['titlu'][$i].'</td>';
 print '<td>'.$_SESSION['autor'][$i].'</td>';
 print '<td>'.$_SESSION['pret'][$i].'</td>';
```



Folosind formulare putem modifica valorile stocate deja în sesiune.

```
 print '<td><input type="text"
   name="nr_buc['.$i.']"
   value="'.$_SESSION['nr_buc'][$i].'">
 </td>';
```

/* calculăm şi cât este totalul pentru această carte, pentru Poezii va fi 1 buc x 100000 lei = 100000 lei în timp ce pentru Manualul de geografie va fi 30 buc x 50000 lei = 1500000 lei*/

```
 $total = $_SESSION['nr_buc'][$i] *
   $_SESSION['pret'][$i];
 print '<td>'.$total.'</td>';
 print '</tr>';
```

/* înainte de a încheia loop-ul, vom adăuga această valoare la totalul general. Variabila $totalGeneral va fi iniţializată la prima rulare a loop-ului şi va avea valoarea totalului primei cărţi. La a doua rulare a loop-ului va avea această valoare plus valoarea totalului celei de-a doua cărţi şi aşa mai departe până când se termină loop-ul. Atunci valoarea sa va fi suma totalurilor. */

```
  $totalGeneral = $totalGeneral +
    $total;
}
print '<table>';
?>
  <input type="submit"
    name="butonulModifica"
    value="Modifică">
</form>
<?
print 'Total General: '.$totalGeneral;
?>
```

Să recapitulăm, folosind pseudocod, execuţia scriptului nostru. El este în momentul de faţă împărţit în cinci părţi distincte:

- pornirea/reiniţializarea sesiunii;
- codul pentru adăugarea de elemente în variabilele de sesiune;
- codul pentru modificarea numărului de cărţi;
- afişarea formularului pentru adăugare de cărţi;
- afişarea listei de cărţi şi a formularului pentru modificare.

La prima rulare a scriptului, codul va fi executat astfel:

- se porneşte sesiunea;
- se trece peste codul de adăugare deoarece nu a fost apăsat butonul Adaugă (nu există variabila $_POST['butonulAdauga']);
- se trece şi peste codul de modificare deoarece nu a fost apăsat butonul Modifică (nu există variabila $_POST['butonulModifica']);
- se afişează formularul;
- se afişează lista de cărţi;

Dacă utilizatorul a completat formularul pentru adăugare, scriptul este rulat astfel:

- se reiniţializează sesiunea;
- se execută codul de adăugare (există variabila $_POST['butonulAdauga']);
- se trece peste codul de modificare deoarece nu a fost apăsat butonul „Modifică" (nu a fost transmisă variabila $_POST['butonulModifica'] deoarece este în alt formular!);
- se afişează formularul;
- se afişează lista de cărţi, inclusiv ultima care tocmai a fost introdusă. Pentru câmpul pret avem un element de tip input care specifică indexul numeric al array-ului, astfel:
- pentru prima carte:

```
<input type="text" name="nr_buc[0]"
  value="10">
```

- pentru a doua carte:

```
<input type="text" name="nr_buc[1]"
  value="5">
```

şi a treia carte:

```
<input type="text" name="nr_buc[2]"
  value="34">
```

Indexul numeric dintre parantezele lui nr_buc[] este acelaşi cu cel din vari-

abila de sesiune $_SESSION['nr_buc'][] şi astfel va fi util pentru a actualiza valorile stocate în sesiune.

Să presupunem că utilizatorul modifică numărul de bucăţi pentru prima carte, şi în loc să cumpere 10, cumpără doar 7. Atunci când apasă butonul „Modifică", execuţia scriptului va fi:

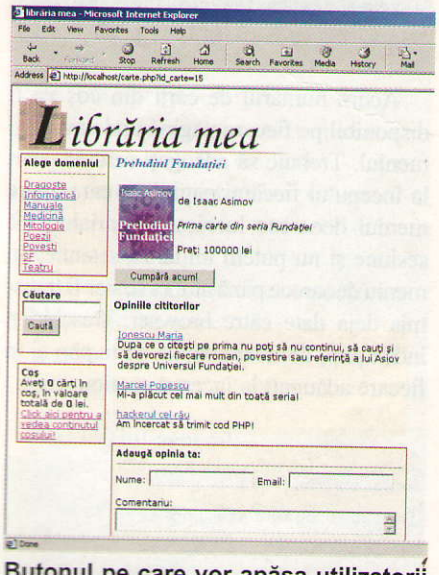
- se reiniţializează sesiunea;
- se trece peste codul de adăugare (nu există variabila $_POST['butonulAdauga']);
- se execută codul pentru modificare. Din formular ne-au fost transmise valorile pentru trei cărţi, $_POST['nr_buc'][0]=7 (modificat în formular), $_POST['nr_buc'][1]=5 şi $_POST['nr_buc'][2]=34.

Trecem prin fiecare element al array-ului folosind for şi actualizăm toate valorile, modificate sau nu (este mai uşor aşa decât să verificăm care au fost modificate şi să le actualizăm doar pe acelea). Atunci:

```
$_SESSION['nr_buc'][0]=$_POST['nr_buc'][0];
// 7, se modifică
$_SESSION['nr_buc'][1]=$_POST['nr_buc'][1];
// 5, nu se modifică dar este actualizat oricum
$_SESSION['nr_buc'][2]=$_POST['nr_buc'][2];
// 34, nu se modifică dar este actualizat oricum
```

- se afişează formularul de adăugare;
- se afişează lista de cărţi cu modificările efectuate.

Cel mai indicat ar fi să scrieţi chiar voi scriptul pentru test.php, să îl rulaţi, să vă uitaţi în sursa documentului HTML (View Source) din când în când (mai ales la formularul de modificare!).



Butonul pe care vor apăsa utilizatorii atunci când doresc să cumpere.

La cumpărături

Pentru coşul nostru de cumpărături vom folosi aproape acelaşi cod, cu singura diferenţă că formularul pentru adăugarea cărţii în listă se va afla în pagina de carte. Deschideţi carte.php şi adăugaţi următorul formular, între informaţiile despre carte şi opiniile utilizatorilor:

```
<form action=
  "cos.php?actiune=adauga"
  method="POST">
<INPUT type="hidden"
  name="id_carte"
  value="<?=$id_carte?>">
<INPUT type="hidden"
  name="titlu" value="
  <?=$rowCarte['titlu']?>">
<INPUT type="hidden"
  name="nume_autor" value=
  "<?=$rowCarte['nume_autor']?>">
<INPUT type="hidden"
  name="pret" value=
  "<?=$rowCarte['pret']?>">
<INPUT type="submit"
  value="Cumpără acum!">
</form>
```

Acest formular va arăta un singur buton pe pagină, „Cumpără acum", dar trimite şi informaţiile care ne interesează, din câmpurile ascunse (hidden) prin metoda POST. Pe lângă acestea va mai trimite şi o variabilă de tip GET, $_GET['actiune'] = "adauga" (din URL-ul action="cos.php?actiune=adauga"). Vom folosi această variabilă pentru a delimita codul de adăugare a informaţiilor în variabile de sesiune. Similar, în formularul de modificare a cărţilor din listă vom avea action="cos.php?actiune=modifica". Iată codul pentru cos.php:

cos.php

```
<?
session_start();
include("conectare.php");
include("page_top.php");
include("meniu.php");
$actiune = $_GET['actiune'];
```

/* Dacă este setată variabila $_GET['actiune'] şi valoarea acesteia este "adauga", se execută acest cod: */

```
if(isset($_GET['actiune']) &&
  $_GET['actiune'] == "adauga")
{
  $_SESSION['id_carte'][]=$_POST['id_carte'];
  $_SESSION['nr_buc'][] = 1;
  $_SESSION['pret'][]=$_POST['pret'];
  $_SESSION['titlu'][]=$_POST['titlu'];
  $_SESSION['nume_autor'][]
     = $_POST['nume_autor'];
}
```

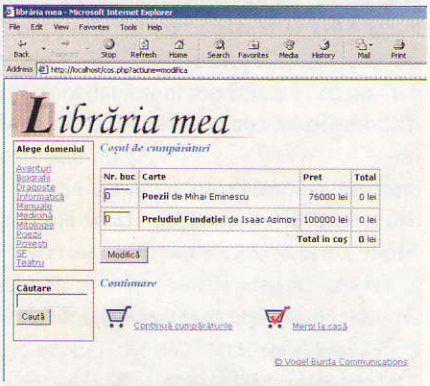
/* Dacă este setată variabila $_GET['actiune'] şi valoarea acesteia este „modifica", se execută acest cod: */

```
if(isset($_GET['actiune']) &&
  $_GET['actiune'] == "modifica")
{
 for($i=0;$i<count($_SESSION['id_carte']);
   $i++)
 {
   $_SESSION['nr_buc'][$i] =
    $_POST['noulNrBuc'][$i];
 }
}
?>
<td valign="top">
<h1>Coşul de cumpărături</h1>
<FORM action="cos.php?actiune=modifica"
  method="POST">
<table border="1" cellspacing="0"
  cellpadding="4">
 <tr bgcolor="#F9F1E7">
  <td><b>Nr. buc</b></td>
  <td><b>Carte</b></td>
  <td><b>Pret</b></td>
  <td><b>Total</b></td>
 </tr>
<?
for($i = 0; $i <
  count($_SESSION['id_carte']); $i++)
{
  print '<tr><td><input type="text"
   name="noulNrBuc['.$i.']" size="1"
   value="'.$_SESSION['nr_buc'][$i].'">
   </td>
   <td><b>'.$_SESSION['titlu'][$i].'</b>
   de '.$_SESSION['nume_autor'][$i].'</td>
  <td align="right">'
   .$_SESSION['pret'][$i].' lei</td>
  <td align="right">'
   .($_SESSION['pret'][$i] *
     $_SESSION['nr_buc'][$i]).
   ' lei</td>
  </tr>';
  $totalGeneral = $totalGeneral +
  ($_SESSION['pret'][$i] *
  $_SESSION['nr_buc'][$i]);
}
```

// şi totalul general:

```
print '<tr><td align="right"
  colspan="3"><b>Total în coş</b></
  td><td align="right"><b>'.$totalGeneral.
  '</b> lei</td></tr>';
?>
</table>
```

Coşul de cumpărături.



Cărţile „şterse" din coşul de cumpărături sunt încă afişate, chiar dacă totalul este 0.

```
<input type="submit"
  value="Modifică"><br><br>
Introduceţi <b>0</b> pentru cărţile
ce doriţi să le scoateţi din coş!
<h1>Continuare</h1>
<table>
  <tr><td width="200" align="center">
  <img src="cos.gif">
  <a href="index.php">Continuă
  cumpărăturile</a></td>
  <td width="200" align="center">
  <img src="casa.gif">
  <a href="casa.php">Mergi la casă</a>
  </td></tr>
</table>
</td>
<?
include("page_bottom.php");
?>
```

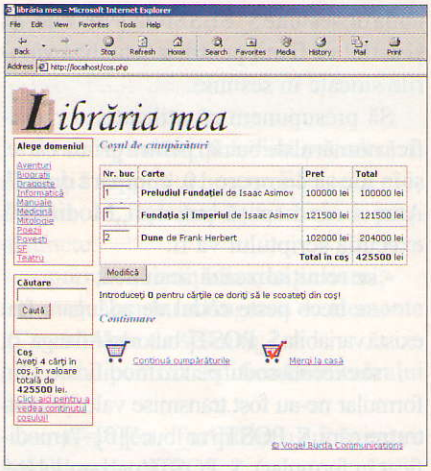
Navigaţi prin paginile librăriei virtuale, adăugaţi cărţi în coş, modificaţi numărul de cărţi pentru a vedea cum funcţionează. Mai avem o problemă de rezolvat: cum scoatem cărţi din coş? Putem pune numărul de bucăţi ca 0 pentru cartea pe care nu dorim să o cumpărăm însă cartea tot apare în listă. Putem condiţiona afişarea cărţilor din loop-ul for: fiecare rând este afişat doar dacă numărul de bucăţi nu este 0. În consecinţă vom modifica loop-ul pentru a specifica şi condiţia de afişare:

```
for($i = 0; $i <
  count($_SESSION['id_carte']); $i++)
{
  if($_SESSION['nr_buc'][$i] != 0)
  /* doar dacă numărul de bucăţi nu e
  0, afişează rândul*/
  {
    print '<tr>
    <td><input type="text"
      name="noulNrBuc['.$i.']"
      size="1"
      value="'.$_SESSION['nr_buc'][$i].'">
    </td>
    <td><b>'.$_SESSION['titlu'][$i].'
    </b> de
    '.$_SESSION['nume_autor'][$i].'
    </td>
    <td align="right">
    '.$_SESSION['pret'][$i].' lei</td>
    <td align="right">'
      .($_SESSION['pret'][$i] *
        $_SESSION['nr_buc'][$i]).
      ' lei</td></tr>';
    $totalGeneral = $totalGeneral +
      ($_SESSION['pret'][$i] *
      $_SESSION['nr_buc'][$i]);
  }
}
```

Acum cărţile cu 0 ca număr de bucăţi nu vor mai fi afişate.

Spuneam la începutul acestui Special că vom pune pe toate paginile site-ului o casetă care să conţină valoarea însumată a cărţilor din coş, pentru o mai bună orientare a utilizatorului. Această casetă o vom pune în meniul din stânga. Deschideţi meniu.php şi adăugaţi următorul cod înainte de </td>:

Numărul şi valoarea cărţilor din coş sunt afişate tot timpul deoarece păstrăm aceste informaţii în variabile de sesiune.



Informaţiile afişate în pagina coşului de cumpărături sunt consistente cu cele afişate în meniu

```
<br>
<div style="width:120px; background-
color:#F9F1E7; padding:4px; border:
solid #632415 1px">
<b>Coş</b><br>
<?
$nrCarti = 0;
$totalValoare = 0;
for($i = 0; $i <
  count($_SESSION['titlu']); $i++)
{
  $nrCarti = $nrCarti +
  $_SESSION['nr_buc'][$i];
  $totalValoare = $totalValoare +
    ($_SESSION['nr_buc'][$i] *
    $_SESSION['pret'][$i]);
}
?>
Aveţi <b><?=$nrCarti?></b> cărţi în
coş, în valoare totală de
<b><?=$totalValoare?></b> lei.
<A href="cos.php">Click aici pentru
a vedea conţinutul coşului!</a>
</div>
```

Acum numărul de cărţi din coş va fi disponibil pe fiecare pagină unde avem şi meniul. Trebuie să adăugaţi session_start la începutul fiecărei pagini pe care apare meniul deoarece lucrăm cu variabile de sesiune şi nu putem iniţializa sesiunea în meniu deoarece până atunci se vor fi transmis deja date către browser. Deschideţi index.php, domeniu.php şi carte.php şi în fiecare adăugaţi la început session_start:

```
<?
session_start();
include("conectare.php");
include("page_top.php");
...
```

La casă

Mai avem de făcut un singur lucru: plata. Pentru aceasta vom face o pagină unde utilizatorul completează un formular cu numele şi adresa unde doreşte să primească comanda şi apoi altă pagină care verifică dacă toate câmpurile formularului au fost completate, introduce informaţiile în baza de date, trimite administratorului un email şi afişează utilizatorului un mesaj de mulţumire. Să facem întâi formularul:

casa.php

```
<?
session_start();
include("conectare.php");
include("page_top.php");
include("meniu.php");
?>
<td valign="top">
<h1>Casa</h1>
Acestea sunt cărţile comandate de
dvs.:
<table border="1" cellspacing="0"
cellpadding="4">
<tr bgcolor="#F9F1E7">
<td><b>Nr. buc</b></td>
<td><b>Carte</b></td>
<td><b>Pret</b></td>
<td><b>Total</b></td>
</tr>
<?
for($i = 0; $i <
count($_SESSION['id_carte']); $i++)
{
 if($_SESSION['nr_buc'][$i] != 0)
 {
 print '<tr><td>'.
 $_SESSION['nr_buc'][$i].'</td>
 <td><b>'.$_SESSION['titlu'][$i].'
 </b> de '.$_SESSION['nume_autor'][$i].'
 </td>
 <td align="right">'
  .$_SESSION['pret'][$i].' lei</td>
 <td align="right">'
  .($_SESSION['pret'][$i] *
 $_SESSION['nr_buc'][$i]).' lei
 </td></tr>';
 $totalGeneral = $totalGeneral +
  ($_SESSION['pret'][$i] *
  $_SESSION['nr_buc'][$i]);
 }
}
```

// şi totalul general:

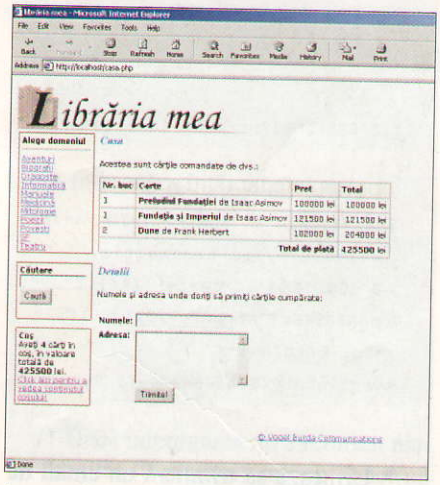
```
print '<tr>
<td align="right" colspan="3">
<b>Total de plată</b></td>
<td align="right">
<b>'.$totalGeneral.'</b> lei</td>
</tr>';
?>
</table>
<h1>Detalii</h1>
```

Numele şi adresa unde doriţi să primiţi cărţile cumpărate:

```
<form action="prelucrare.php"
  method="POST">
<table>
<tr>
 <td><b>Numele:</b></td>
 <td><input type="text"
  name="nume"></td>
</tr>
<tr>
 <td valign="top"><b>Adresa:</b></td>
 <td><textarea name="adresa"
  rows="6"></textarea></td>
</tr>
<tr>
 <td></td>
 <td><INPUT type="submit"
  value="Trimite!"></td>
</tr>
</table>
</form>
</td>
<?
include("page_bottom.php");
?>
```

Fişierul care prelucrează aceste informaţii, prelucrare.php, va executa următoarele acţiuni:

- va verifica dacă numele este completat;
- va verifica dacă adresa este completată;
- va verifica numărul cărţilor din coş astfel încât utilizatorul să nu trimită formularul fără să fi cumpărat vreo carte;
- se va conecta la baza de date;
- va introduce o nouă înregistrare în tabelul tranzactii cu numele, adresa şi data tranzacţiei;
- va lua id_tranzactie corespunzător acestei înregistrări;
- va folosi acest id_tranzactie pentru a introduce în tabelul vânzări cărţile comandate;
- vă va trimite un email de notificare;
- „curăţă" sesiunea, ştergând toate datele salvate în ea deoarece nu mai e nevoie de ele şi ar putea induce confuzie în utilizator;
- afişează utilizatorului pagina, cu un mesaj de mulţumire.



La casă: numele şi adresa unde utilizatorul va primi cărţile comandate.

Înainte de a trece la scrierea codului, trebuie să clarificăm un lucru: funcţia mail. Pe sistemele Linux aceasta foloseşte sendmail sau qmail pentru a trimite mesaje, pe sistemele Windows foloseşte SMTP. Dacă sunteţi pe o maşină Windows 2000 sau XP puteţi porni serviciul SMTP pentru a trimite emailuri dar dacă aveţi Windows 98, 95 sau NT va trebui să modificaţi setările din php.ini (din directorul Windows) şi să folosiţi SMTP-ul ISP-ului pentru a trimite emailuri (cel pe care îl folosiţi şi în clientul de mail). Singurul inconvenient pentru trimiterea de emailuri prin SMTP-ul ISP-ului este că trebuie să fiţi conectaţi la Internet, altfel veţi primi un mesaj de eroare. Iată cum arată setarea SMTP în php.ini la mine:

```
[mail function]
; For Win32 only.
SMTP = mail.isp.ro
```

Dacă sunteţi pe o maşină Windows 2000 sau XP puteţi seta SMTP = localhost, dar serviciul de SMTP trebuie să fie pornit! Faceţi modificarea în php.ini apoi faceţi un fişier de test, numit mail.php:

mail.php

```
<?
mail("oana@chip.ro","test mesaj
 smtp","testing");
print "ok";
?>
```

Înlocuiţi adresa mea de email cu a dvs. (dvs. trebuie să primiţi emailul, nu eu) şi rulaţi scriptul în browser. Dacă nu sunteţi conectaţi la Internet, adresa serverului SMTP este greşită sau serviciul SMTP nu este pornit la adresa respectivă, veţi primi un mesaj de eroare, altfel veţi primi mesajul la adresa dvs. de email.

Desigur, nu este foarte convenabil să trebuiască să fiţi conectaţi la Internet de câte ori doriţi să trimiteţi un emai prin

intermediul PHP.

Din fericire, avem o soluție simplă pentru această problemă: putem pune operatorul @ în fața funcției mail și astfel împăcăm și capra și varza: dacă poate trimite mail îl trimite, dacă nu, nu returnează nici un mesaj de eroare (dar nici nu va trimite emailul).

mail.php

```
<?
@mail("oana@chip.ro","test mesaj
  smtp","testing");
print "ok";
?>
```

Astfel putem lucra și testa în continuare aplicația offline, iar dacă la un moment dat dorim să o transferăm pe un server care poate trimite email, nu va mai trebui să facem nici un fel de modificări.

Cu acestea lămurite, să scriem codul pentru fișierul prelucrare.php

prelucrare.php

```
<?
```

// Verificăm dacă numele este completat, iar dacă nu e, oprim execuția scriptului

```
if($_POST['nume'] == "")
{
  print 'Trebuie să completați numele!
    <a href="cos.php">Înapoi</a>';
  exit;
}
```

/* Verificăm dacă adresa este completată iar dacă nu e, oprim execuția scriptului */

```
if($_POST['adresa'] == "")
{
  print 'Trebuie să completați adresa!
    <a href="cos.php">Înapoi</a>';
  exit;
}
```

/* Reinițializăm sesiunea deoarece dorim să verificăm numărul de cărți comandate */

```
session_start();
```

/* numărul de cărți comandate îl aflăm rapid, folosind array_sum. array_sum($array) returnează suma valorilor dintr-un array dacă acestea sunt numerice. Astfel, dacă $a = array('1','1','2'), array_sum($a) = 4. Nu confundați array_sum cu count, count($a) = 3 elemente în timp ce array_sum($a) = 4 (suma elementelor). */

```
$nrCarti=array_sum($_SESSION['nr_buc']);
if($nrCarti == 0)
{
  print 'Trebuie să cumpărați cel
    puțin o carte! <a
    href="cos.php">Înapoi</a>';
  exit;
}
```

/* În acest moment toate datele sunt verificate, putem să ne conectăm la baza de date pentru a le introduce: */

```
include("conectare.php");
```

/* Introducem întâi datele în tabelul tranzactii. Deoarece câmpul data din tabel este de tip TIMESTAMP, îl putem omite (se va seta singur, cu data curentă) */

```
$sqlTranzactie = "insert into
  tranzactii(nume_cumparator,
  adresa_cumparator)
  values('".$_POST['nume'].
  "','".$_POST['adresa']."')";
$resursaTranzactie =
  mysql_query($sqlTranzactie);
```

/* Obținem id_ul acestei înregistrări folosind mysql_insert_id: */

```
$id_tranzactie = mysql_insert_id();
```

/* iar acum luăm fiecare carte din sesiune și o introducem în tabelul vânzări. Introducem în tabel doar cărțile al căror număr de bucăți este mai mare ca 0 (în condiția if, din cadrul structurii for): */

```
for($i=0;$i<count($_SESSION['id_carte']);
$i++)
{
  if($_SESSION['nr_buc'][$i] > 0)
  {
    /* Creăm interogarea */
    $sqlVanzare = "INSERT INTO vanzari
      values('".$id_tranzactie."','".
      $_SESSION['id_carte'][$i]."','".
      $_SESSION['nr_buc'][$i]."')";
    /* și o rulăm */
    mysql_query($sqlVanzare);
  }
}
```

/* Urmează să trimitem un email de notificare folosind funcția mail. mail folosește în principal trei argumente: mail(adresa destinatarului, subiectul mesajului, textul mesajului) dar mai poate prelua încă unul, pentru headere adiționale. */

```
$emailDestinatar = "oana@chip.ro";
```

/* schimbați adresa cu cea la care doriți să primiți mesajele */

```
$subiect = "O nouă comandă!";
```

/* Pentru a compune mesajul ne vom folosi de operatorul .= de concatenare a stringurilor. */

```
$mesaj = "O nouă comandă de la
  <b>".$_POST['nume']."</b><br>";
$mesaj .= "Adresa:
  ".$_POST['adresa']."<br>";
$mesaj .= "Cartile
  comandate:<br><br>";
$mesaj .= "<table border='1'
  cellspacing='0' cellpadding='4'>";
for($i=0;$i<count($_SESSION['id_carte']);
$i++)
{
  if($_SESSION['nr_buc'][$i] > 0)
  {
    $mesaj .= "<tr><td>"
    .$_SESSION['titlu'][$i]." de
    ".$_SESSION['nume_autor'][$i].
    "</td><td>".$_SESSION['nr_buc'][$i].
    " buc</td></tr>";
    $totalGeneral = $totalGeneral +
     ($_SESSION['nr_buc'][$i] *
     $_SESSION['pret'][$i]);
  }
}
$mesaj .= "</table>";
$mesaj .= "Total:
  <b>".$totalGeneral."</b>";
```

/* Punem headere adiționale pentru a trimite mesajul în format HTML și encodingul potrivit pentru caracterele românești: */

```
$headers = "MIME-Version:
  1.0\r\nContent-type: text/html;
  charset=iso-8859-2\r\n";
```

/* Acum putem trimite emailul: */

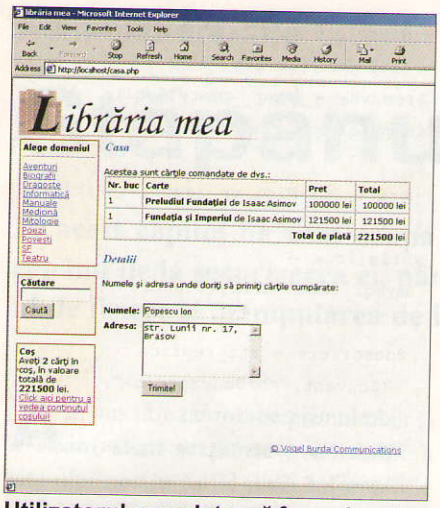
```
mail($emailDestinatar, $subiect,
  $mesaj, $headers);
```

/* Curățăm sesiunea deoarece nu mai avem nevoie de datele din ea.*/

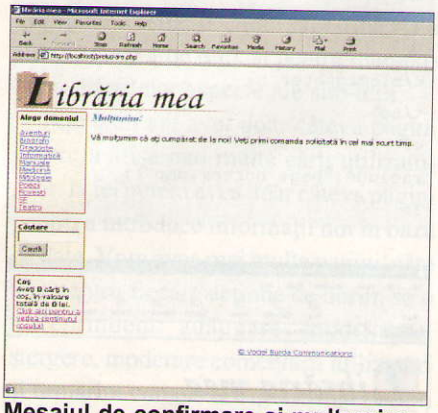
```
session_unset();
```

/* eliminăm toate variabilele asociate acestei sesiuni */

```
session_destroy();
```



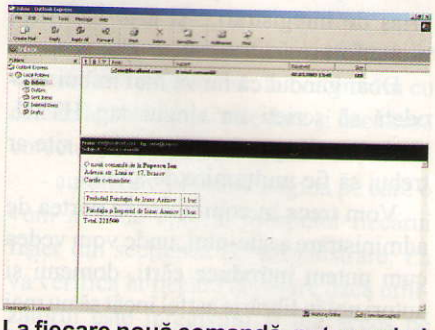
Utilizatorul completează formularul cu informațiile personale.



Mesajul de confirmare şi mulțumire

/* ştergem sesiunea */
/* În final afişăm utilizatorului pagina cu mesajul de mulțumire: */

```
include("page_top.php");
include("meniu.php");
?>
<td valign="top">
<h1>Mulțumim!</h1>
Vă mulțumim că ați cumpărat de la
noi! Veți primi comanda solicitată
în cel mai scurt timp.
</td>
<?
include("page_bottom.php");
?>
```



La fiecare nouă comandă, putem primi email de notificare.

Căutare

Mai avem o singură pagină de făcut: cea de căutare. Vom lăsa posibilitatea utilizatorului să caute în baza noastră de date şi să îi afişeze rezultatele pe o pagină. Să deschidem întâi meniu.php unde se află formularul pentru căutare. Observați că are metoda GET (<form action="cautare.php" method="GET">). De ce? Pentru că atunci vom putea adăuga în lista Favorites a browserului rezultatele unei căutări. Cu metoda POST, adresa paginii de căutare după apăsarea butonului "Caută" ar fi http://localhost/cautare.php dar cu GET ea va fi http://localhost/cautare.php?cuvant=poezii. Acest URL îl putem apoi trimite altcuiva pentru a vedea direct rezultatele căutării sau îl putem adăuga la lista de Favorites pentru mai târziu.

Înainte de a scrie codul, să ne aducem aminte care era comanda SQL pentru selectarea parțială a unui cuvânt. Foloseam wildcard-ul %, astfel: SELECT * FROM table WHERE coloana LIKE "%cuvant%". Aceeaşi comandă SQL o vom folosi şi în continuare pentru a interoga tabelele carti şi autori după cuvântul sau cuvintele scrise de utilizator:

```
<?
session_start();
include("conectare.php");
include("page_top.php");
include("meniu.php");
$cuvant = $_GET['cuvant'];
?>
<td valign="top">
<h1>Rezultatele căutării</h1>
<p>Textul căutat: <b><?=$cuvant?>
</b></p>
<b>Autori</b>
<blockquote>
```

/* Interogăm întâi tabelul autori:*/

```
<?
$sql = "SELECT id_autor, nume_autor
  FROM autori WHERE nume_autor LIKE
  '%".$cuvant."%'";
$resursa = mysql_query($sql);
```

/* Dacă interogarea nu a returnat nici un rezultat (0 rânduri) scriem "Nici un rezultat" */

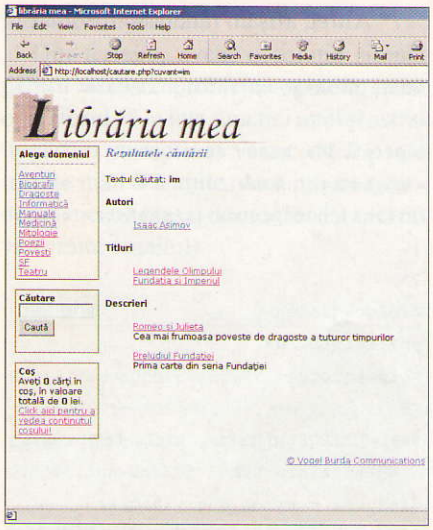
```
if(mysql_num_rows($resursa) == 0)
{
 print "<i>Nici un rezultat</i>";
}
```

/* Altfel, afişăm rândurile rezultate*/

```
while($row = mysql_fetch_array($resursa))
{
 print '<a href="autor.php?
 id_autor='.$row['id_autor'].'">'
 .$row['nume_autor'].'</a><br>';
}
?>
</blockquote>
<b>Titluri</b>
<blockquote>
<?
$sql="SELECT id_carte, titlu FROM carti
  WHERE titlu LIKE '%".$cuvant."%'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) == 0)
{
 print "<i>Nici un rezultat</i>";
}
while($row=mysql_fetch_array($resursa))
{
 print '<a href="carte.php?id_carte='
 .$row['id_carte'].'">'.$row['titlu'].'
 </a><br>';
}
?>
</blockquote>
<b>Descrieri</b>
<blockquote>
<?
$sql = "SELECT id_carte, titlu,
 descriere FROM carti WHERE
 descriere LIKE '%".$cuvant."%'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) == 0)
{
 print "<i>Nici un rezultat</i>";
}
while($row=mysql_fetch_array($resursa))
{
 print '<a
 href="carte.php?id_carte='
 .$row['id_carte'].'">'.$row['titlu'].
 '</a><br>'.$row['descriere'].'<br><br>';
}
?>
</blockquote>
</td>
<?
include("page_bottom.php");
?>
```

Observați că am făcut referire la un fişier inexistent, autori.php. Aceasta va fi tema voastră. Un click pe numele lui Mihai Eminescu în pagina de căutare ne va duce la pagina http://localhost/autor.php?id_autor=1.

Folosiți $_GET['id_autor'] pentru a interoga tabelul cărți şi a obține o listă cu



Rezultatele căutării pe site

cărțile scrise de autorul respectiv. Luați pagina domeniu.php ca exemplu.

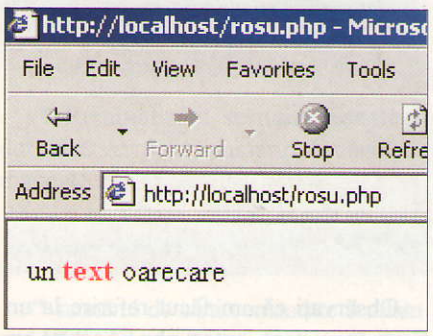
Să revenim la căutarea noastră. Cum am putea afișa rezultatele astfel încât cuvântul sau textul căutat să fie evidențiate, la fel ca în Google? Putem folosi funcția str_replace pentru a înlocui o parte dintr-un string. Funcția se folosește în felul următor:

```
$stringModificat = str_replace
  ("textul vechi din string ce urmează a fi modificat", "textul nou,
  modificat", $stringVechi);
```

Iată într-un mic exemplu, cum se folosește această funcție:

rosu.php

```
<?
$text = "un text oarecare";
$text = str_replace("text","<b><font
  color='red'>text</font></b>",$text);
print $text;
?>
```

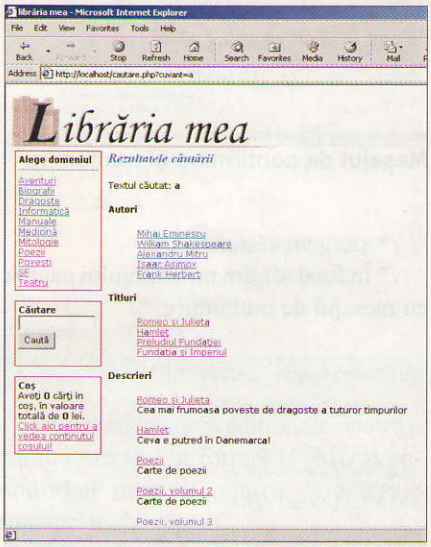


Exemplu str_replace.

Aceeași funcție o vom folosi și înainte de a afișa rezultatele căutării, pentru a evidenția (bold) textul căutat. Iată fișierul cautare.php modificat astfel încât să fie evidențiat textul căutat:

```
<?
session_start();
include("conectare.php");
include("page_top.php");
include("meniu.php");
$cuvant = $_GET['cuvant'];
?>
<td valign="top">
<h1>Rezultatele căutării</h1>
<p>Textul căutat: <b><?=$cuvant?>
</b></p>
<b>Autori</b>
<blockquote>
<?
$sql = "SELECT id_autor, nume_autor
  FROM autori WHERE nume_autor LIKE
  '%".$cuvant."%'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) == 0)
{
  print "<i>Nici un rezultat</i>";
}
while($row =
  mysql_fetch_array($resursa))
{
  $nume_autor = str_replace
    ($cuvant,"<b>$cuvant</b>",
    $row['nume_autor']);
  print '<a href="autor.php?
    id_autor='.$row['id_autor'].'">
    '.$nume_autor.'</a><br>';
}
?>
</blockquote>
```

```
<b>Titluri</b>
<blockquote>
<?
$sql="SELECT id_carte, titlu FROM carti
  WHERE titlu LIKE '%".$cuvant."%'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) == 0)
{
  print "<i>Nici un rezultat</i>";
}
while($row =
  mysql_fetch_array($resursa))
{
  $titlu = str_replace
    ($cuvant,"<b>$cuvant</b>",
    $row['titlu']);
  print '<a href="carte.php?
    id_carte='.$row['id_carte'].'">'
    .$titlu.'</a><br>';
}
?>
</blockquote>
<b>Descrieri</b>
<blockquote>
<?
$sql = "SELECT id_carte, titlu,
  descriere FROM carti WHERE descriere
  LIKE '%".$cuvant."%'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) == 0)
{
  print "<i>Nici un rezultat</i>";
}
while($row =
  mysql_fetch_array($resursa))
{
  $descriere = str_replace
    ($cuvant,"<b>$cuvant</b>",
    $row['descriere']);
  print '<a href="carte.php?id_carte='
  .$row['id_carte'] .'">'
  .$row['titlu'].'</a><br>
  '.$descriere.'<br><br>';
}
?>
</blockquote>
</td>
<?
include("page_bottom.php");
?>
```



Textul căutat este evidențiat în rezultatele căutării.

Cu acestea am terminat partea „vizitabilă" a site-ului nostru. După cum v-am promis la început, am făcut doar câteva fișiere care să poată afișa un număr uriaș de înregistrări... și asta fără prea mult efort.

Doar gândul că nu va mai trebui vreodată să scrieți un singur tag HTML pentru a adăuga elemente noi pe site ar trebui să fie mulțumire de ajuns!

Vom trece în continuare la partea de administrare a site-ului, unde vom vedea cum putem introduce cărți, domenii și autori noi în librărie astfel încât să nu mai avem vreodată de-a face cu chinuitoarea linie de comandă.

Administrare site

Stăpânul datelor

În acest capitol ne vom ocupa de secțiunea de administrare, începând de la securizarea ei, până la introducerea datelor, upload-ul de fișiere și manipularea de imagini.

Am pus utilizatorul pe primul plan și am făcut secțiunea „vizitabilă" a site-ului nostru astfel încât informațiile să fie întotdeauna „la zi". Este momentul să ne gândim și la noi și la timpul pe care l-am putea câștiga dacă am avea o interfață simplă și ușor de utilizat pentru manipularea diferitelor aspecte ale site-ului.

Așa cum am avut doar câteva pagini pentru a afișa mai multe cărți utilizatorilor, la fel putem avea doar câteva pagini pentru a introduce informații noi în baza de date. Vom avea mai multe pagini, câte una pentru fiecare acțiune ce dorim să o intreprindem: adăugare, modificare, ștergere, moderare comentarii utilizatori și urmărire comenzi. Vom învăța și cum să oferim acces în această zonă doar administratorului sau altor utilizatori autorizați.

Creați un director nou în document root și denumiți-l administrare. Aici vom păstra toate fișierele secțiunii de administrare, ele fiind accesibile la adresa http://localhost/administrare/

Autentificare

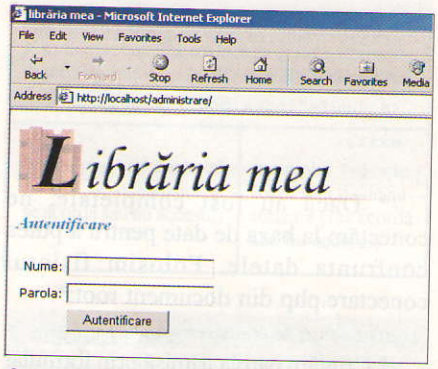
Primul lucru pe care trebuie să îl facem este să asigurăm această zonă, să nu permitem accesul decât administratorului, cu o anumită combinație de nume și parolă. Pentru aceasta vom scrie trei fișiere:

- index.php unde vom afișa formularul de login
- login.php unde verificăm informațiile transmise din formular, le coroborăm cu cele existente în baza de date și dacă sunt corecte, acordăm accesul mai departe
- autorizare.php este pagina pe care o vom include apoi la începutul fiecărui fișier din secțiunea de administrare. Ea va verifica la fiecare accesare dacă utilizatorul este înregistrat și are acces pe pagina respectivă și permite rularea paginii doar dacă utilizatorul este înregistrat. Fără această verificare, un utilizator ar putea accesa http://localhost/pagina.php, fără să treacă prin formularul de înregistrare.

Să trecem la treabă. După cum spuneam, index.php conține un simplu formular:

administrare/index.php

```
<html>
 <head>
 <meta http-equiv="Content-Type"
 content="text/html; charset=iso-8859-2">
 <title>librăria mea</title>
 <style type="text/css">
  body, p, td {font-family: Verdana,
  Arial, sans-serif; font-size:
  12px;}
  h1 {font-family: Times New Roman,
  Times, serif; font-size: 18px;
  font-weight: bold; color: #336699;
  font-style:italic;}
  .titlu {font-family: Verdana, Ari-
  al, sans-serif; font-size: 14px;
  font-weight: bold; color:
  #0066CC;}
 </style>
</head>
<body bgcolor="#ffffff">
<img src="../logo.gif">
<h1>Autentificare</h1>
<FORM action="login.php"
method="POST">
 <table>
  <tr>
   <td align="right">Nume: </td>
   <td><INPUT type="text"
   name="nume"></td>
  </tr>
  <tr>
   <td align="right">Parola: </td>
   <td><INPUT type="password"
   name="parola"></td>
  </tr>
  <tr>
   <td></td>
   <td><INPUT type="submit"
   value="Autentificare"></td>
  </tr>
 </table>
</form>
</body>
</html>
```



Accesul în secțiunea de administrare se face pe bază de nume și parolă.

Pentru ca în următorul fișier, login.php, să poată confrunta datele din formular cu cele din baza de date, trebuie întâi să avem informațiile stocate în baza de date. Va trebui să facem pentru aceasta un nou tabel, admin, în baza de date librarie:

```
CREATE TABLE admin (
  admin_nume text NOT NULL,
  admin_parola text NOT NULL
) ;
```

În acest tabel vom salva numele și parola pentru accesul la secțiunea de administrare. Dar, cum salvarea parolei „în clar" este în general o idee proastă, o vom introduce în tabel criptată:

```
INSERT INTO admin VALUES
 ("administrator", md5("parola"));
```

Iată cum arată conținutul tabelului:

```
SELECT * FROM admin;
+---------------+------------------------------------+
|admin_nume | admin_parola                           |
+---------------+------------------------------------+
|administrator|8287458823facb8ff918dbfabcd22ccb|
+---------------+------------------------------------+
```

Criptarea folosind md5 nu este reversibilă (și astfel nici dvs., nici altcineva nu o va putea afla chiar dacă are acces la baza de date). Să scriem scriptul care

să confrunte informațiile din formular cu cele din baza de date:

administrare/login.php

```
<?
/* Verificăm întâi dacă au fost completate amândouă câmpurile */
if($_POST['nume'] == "" ||
  $_POST['parola'] == "")
{
  print 'Trebuie să completezi amândouă câmpurile!<br>
  <a href="index.php">Înapoi</a>';
  exit;
}
```

/* Dacă au fost completate, ne conectăm la baza de date pentru a putea confrunta datele. Folosim fișierul conectare.php din document root:*/

```
include("../conectare.php");
```

/* Criptăm parola trimisă prin formular tot cu md5, pentru a o avea în formatul în care ea există în baza de date: */

```
$parolaEncriptata=md5($_POST['parola']);
```

/* şi scriem interogarea de verificare: */

```
$sql = "SELECT * FROM admin WHERE
  admin_nume='".$_POST['nume']."' AND
  admin_parola='".$parolaEncriptata."'";
$resursa = mysql_query($sql);
```

/* Această interogare, dacă este executată cu succes (numele şi parola din formular sunt valide şi corespund celor din baza de date) ar trebui să returneze un singur rând din tabelul admin, care corespunde numelui şi parolei. Pentru mulți programatori, verificarea returnării unui rezultat este de ajuns pentru a acorda accesul mai departe. Noi vom extinde protecția şi vom acorda accesul doar dacă rezultatul returnat are un singur rând (mai precis, vom afişa un mesaj de eroare şi vom opri execuția scriptului dacă numărul de rânduri nu e 1): */

```
if(mysql_num_rows($resursa) != 1)
{
  print 'Nume sau parolă
  greşite!<br>
  <a href="index.php">Înapoi</a>';
  exit;
}
```

/* Până acum s-au făcut toate verificările, numele şi parola sunt corecte, să trecem la autentificarea propriu-zisă. Vom folosi variabile de sesiune pentru a păstra în memorie câteva informații despre autentificare, pentru a le reverifica mai târziu, atunci când accesăm alte pagini din cadrul secțiunii de administrare. Pornim întâi sesiunea după care trecem la salvarea informațiilor în ea: */

```
session_start();
$_SESSION['nume_admin'] =
$_POST['nume'];
$_SESSION['parola_encriptata'] =
      $parolaEncriptata;
```

/* Pe lângă acestea, pentru o şi mai mare siguranță, vom salva id-ul sesiunii în altă variabilă. Toate sesiunile au un id unic, un string care seamănă cu rezultatul unei criptări md5.*/

```
$_SESSION['key_admin'] = session_id();
```

/* Cu autentificarea făcută, spunem scriptului să încarce prima pagină din secțiunea de administrare: */

```
header("location: admin.php");
?>
```

Creați şi o pagină admin.php în care scrieți textul „secțiunea de administrare", după care accesați adresa http://localhost/administrare/ şi autentificați-vă cu numele „administrator" şi parola „parola,,. Autentificarea va fi făcută şi veți fi redirecționați către pagina admin.php. Să facem un alt test: închideți toate instanțele browserului astfel încât să expire sesiunea, porniți din nou un browser şi accesați direct adresa http://localhost/administrare/admin.php. Veți avea acces, chiar dacă nu v-ați autentificat folosind formularul! Ei bine, putem împiedica accesul neautorizat dacă scriem un mic script de verificare a datelor sesiunii înainte de a încărca orice pagină din secțiunea de administrare. Iată codul:

autorizare.php

```
<?
```

/* Pornim sesiunea, deoarece avem nevoie de datele din ea: */

```
session_start();
```

/* şi verificăm datele, începând cu $_SESSION['key_admin'] definit în login.php care trebuie să fie același cu session_id: */

```
if($_SESSION['key_admin']!=session_id())
{
  print 'Acces neautorizat!';
  exit;
}
```

/* Ne conectăm la baza de date deoarece vom reverifica datele din sesiune cu cele din baza de date: */

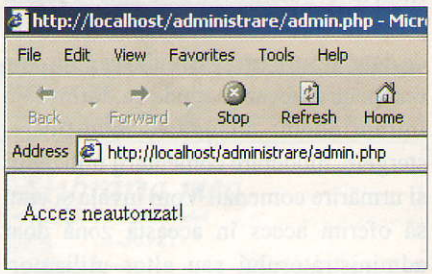
```
include("../conectare.php");
```

/* Verificăm dacă numele şi parola salvate în variabile de sesiune sunt aceleaşi cu cele din baza de date*/

```
$sql = "SELECT * FROM admin WHERE
  admin_nume='".$_SESSION ['nume_admin']."'
  AND admin_parola=
  '".$_SESSION['parola_encriptata']."'";
$resursa = mysql_query($sql);
```

/* Această interogare, dacă este executată cu succes (numele şi parola din variabilele de sesiune sunt valide şi corespund celor din baza de date) ar trebui să returneze un singur rând din tabelul admin. Dacă nu returnează exact un rând vom afişa un mesaj de eroare şi vom opri execuția scriptului: */

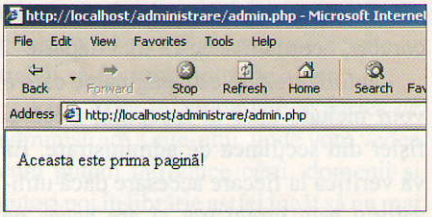
```
if(mysql_num_rows($resursa) != 1)
{
  print 'Acces neautorizat!';
  exit;
}
?>
```



Dacă numele şi parola nu sunt corecte, nu se acordă acces mai departe.

Vom folosi autorizare.php la începutul fiecărei pagini şi împuşcăm astfel trei iepuri dintr-un foc: în acest script inițializăm sesiunea, ne conectăm la baza de date şi verificăm autentificarea. Ca urmare, dacă includem acest script în orice fişier din secțiunea de administrare vom fi şi gata conectați la baza de date, şi cu sesiunea pornită şi cu autorizarea efectuată. Să rescriem fişierul admin.php:

```
<?
include("autorizare.php");
print "Aceasta este prima pagină!";
?>
```



Dacă numele şi parola sunt valide, accesul este acordat.

Închideți toate instanțele browserului pentru a șterge sesiunea (dacă v-ați logat) și apoi accesați adresa http://localhost/administrare/admin.php (direct, fără să treceți prin formularul de autentificare). Vom primi mesajul „Acces neautorizat!" și restul textului nu va mai fi afișat! Accesați acum http://localhost/administrare/ și completați formularul de autentificare. Dacă le-ați completat corect, veți ajunge pe pagina admin.php și veți vedea textul „Aceasta este prima pagină!". Paginile de pe secțiunea de administrare sunt de acum asigurate!

Pentru că zona de administrare are mai multe secțiuni, pentru adăugare, ștergere, etc., va trebui să avem link-uri către toate acele secțiuni, pe fiecare pagină. Decât să le scriem de fiecare dată când avem nevoie de ele, mai bine le scriem într-un fișier, admin_top.php pe care să îl accesăm atunci când avem nevoie și în care scriem și instrucțiunile <html>, <body>, etc. Iată conținutul fișierului:

admin_top.php

```
<html>
<head>
 <meta http-equiv="Content-Type"
 content="text/html; charset=iso-8859-2">
 <title>librăria mea</title>
 <style type="text/css">
 body, p, td {font-family: Verdana,
 Arial, sans-serif; font-size:
 12px;}
 h1 {font-family: Times New Roman,
 Times, serif; font-size: 18px;
 font-weight: bold; color: #336699;
 font-style:italic;}
 .titlu {font-family: Verdana, Ari-
 al, sans-serif; font-size: 14px;
 font-weight: bold; color: #0066CC;}
 </style>
 </head>
<body bgcolor="#ffffff">
<img src="../logo.gif">
<TABLE bgcolor="#F9F1E7" cellspac-
ing="0" cellpadding="4" border="1">
  <tr>
    <td><A
    href="adaugare.php">Adaugă</a></
    td>
    <td><A
    href="modificare_stergere.php">
    Modifică sau șterge</a></td>
    <td><A href="opinii.php">Opinii
    vizitatori</a></td>
    <td><A
    href="comenzi.php">Comenzi</a></
    td>
  </tr>
 </table>
<br>
```

Să modificăm acum și admin.php și să includem acest fișier în el:

admin.php

```
<?
include("autorizare.php");
include("admin_top.php");
print "Aceasta este prima pagină!";
?>
```

Prima pagină a secțiunii de administrare.

Uneori, includerea unui fișier care să se ocupe cu autorizarea nu este o soluție viabilă. De exemplu, putem avea următorul exemplu în care o pagină a secțiunii de administrare (test.php) include alte două pagini(meniu.php și main.php). Atunci ar trebui să includem fișierul autorizare.php în fiecare din ele, dar atunci am primi un mesaj de eroare (se reinițializează sesiunea de 3 ori și în plus, în meniu.php și main.php se reinițializează după ce au fost trimise date către browser).

Atunci putem crea o funcție declarată de noi, care să facă autorizarea.

Această funcție o putem apoi apela oricând avem nevoie de ea (cu condiția să fie definită întâi!).

Iată cum ar arăta structura site-ului în acest caz:

test.php (include autorizare.php care pornește sesiunea, se conectează la baza de date, verifică autorizarea)	
meniu.php (include autorizare.php care pornește sesiunea, se conectează la baza de date, verifică autorizarea)	main.php (include autorizare.php care pornește sesiunea, se conectează la baza de date, verifică autorizarea)

Observăm că în cele două fișiere care sunt parte a lui test.php nu facem decât să verificăm autorizarea, fără să includem nici un alt fișier. Cum funcția de autorizare este definită în fișierul „mamă", dacă meniu.php sau main.php sunt apelate din cadrul lui, vor face verificarea folosind funcția deja definită. Iată structura celor patru fișiere:

test.php (include f_autorizare.php care pornește sesiunea, se conectează la baza de date, definește funcția de autorizare și folosește această funcție pentru a face autorizarea)	
meniu.php (se apelează funcția de autorizare și se acordă sau nu accesul)	main.php (se apelează funcția de autorizare și se acordă sau nu accesul)

f_autorizare.php

```
<?
session_start();
include("../conectare.php");
function autorizat()
{
  $sql = "SELECT * FROM admin WHERE
  admin_nume='".$_SESSION['nume_admin']."'
  AND admin_parola='".$_SESSION
  ['parola_encriptata']."'";
  $resursa = mysql_query($sql);
  if($_SESSION['key_admin'] !=
  session_id() ||
  mysql_num_rows($resursa) != 1)
  {
   return false;
  }
  else
  {
   return true;
  }
}
?>
```

test.php

```
<?
```

/* include fișierul f_autorizare.php care pornește sesiunea, se conectează la baza de date și declară funcția de autorizare*/

```
include("f_autorizare.php");
```

/* și verifică autorizarea folosind funcția definită în fișierul pe care tocmai l-am inclus, f_autorizare.php*/

```
if(!autorizat())
{
  print 'Acces neautorizat!';
  exit;
```

```
}
include("meniu.php");
include("main.php");
?>
```

meniu.php

```
/* verifică doar autorizarea*/
if(!autorizat())
{
  print 'Acces neautorizat!';
  exit;
}
/* codul care re rulează dacă este autorizat */
```

test.php (defineşte funcţia de autorizare, foloseşte această funcţie pentru a face autorizarea)
meniu.php, inclus în test.php (defineşte funcţia de autorizare, foloseşte această funcţie pentru a face autorizarea => eroare! funcţia a fost definită de două ori (o dată în fişierul „mamă", o dată aici.

main.php

```
/* verifică doar autorizarea*/
if(!autorizat())
{
  print 'Acces neautorizat!';
  exit;
}
/* codul care se rulează dacă este autorizat */
```

Astfel vedem cum, folosind o funcţie definită de noi, nu se va putea accesa nici unul din fişierele incluse fără autorizare! De cele mai multe ori, o funcţie este mult mai utilă decât un fişier inclus însă alegerea este a dvs. Fiţi atenţi doar să nu definiţi funcţia de două ori în cadrul aceluiaşi script, altfel veţi primi mesajul de eroare **Fatal error**: Cannot redeclare numele_functiei().

Dacă funcţia a fost deja definită undeva în execuţia scriptului, trebuie doar să o apelaţi.

Adăugare

În baza de date putem adăuga trei lucruri: domenii, autori sau cărţi noi. Pentru acestea vom folosi o pagină în care vom afişa trei formulare, câte unul pentru fiecare şi altă pagină care va prelucra informaţiile din aceste formulare, în funcţie de care a fost trimis către server. În pagina adaugare.php vom avea:

- formularul pentru introducere domeniu nou;
- formularul pentru introducere autor nou;
- formularul pentru introducere carte nouă;

Pagina prelucrare_adaugare.php va avea următoarea structură:

- include pagina admin_top.php (se porneşte sesiunea, se conectează la baza de date şi încarcă începutul paginii);
- dacă a fost trimis primul formular, îl prelucrează şi introduce domeniul nou în tabelul domenii şi apoi afişează un mesaj de confirmare;
- dacă a fost trimis al doilea formular, îl prelucrează, îl introduce noul autor în tabelul autori şi afişează un mesaj de confirmare;
- dacă a fost trimis al treilea formular, îl prelucrează şi introduce noua carte în tabelul carti şi apoi afişează un mesaj de confirmare.

Înainte de a crea cele două fişiere, vom mai face un singur lucru. Ne aducem aminte că tabelul carti are un câmp de tip DATE numit data în care se stochează data adăugării cărţii în baza de date. Această dată ne foloseşte pentru a afişa pe prima pagină cele mai noi trei cărţi. Ştim din experienţă că este mult mai simplu să lucrăm cu câmpuri de tip TIMESTAMP (se setează automat cu data curentă), aşa că ne vom conecta la baza de date şi vom modifica tipul câmpului:

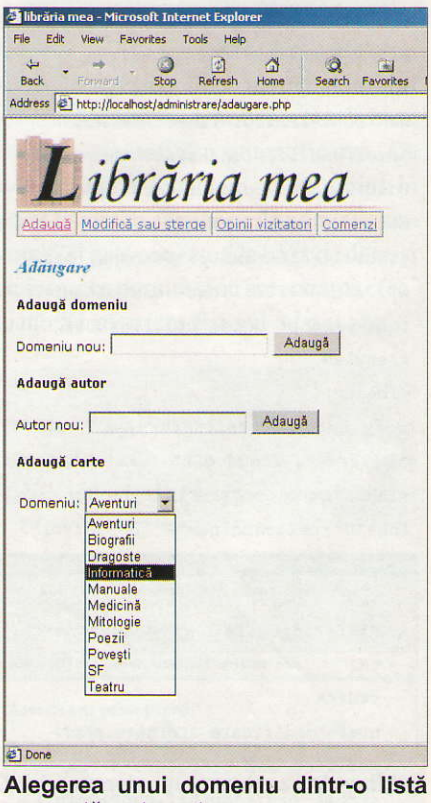
```
ALTER TABLE carti CHANGE data data
TIMESTAMP(10) NOT NULL;
```

Toate valorile înregistrate anterior vor fi convertite, iar de acum încolo câmpul data se va seta automat la fiecare INSERT. Altfel ar fi trebuit să îl specificăm noi.

Iată cele două fişiere:

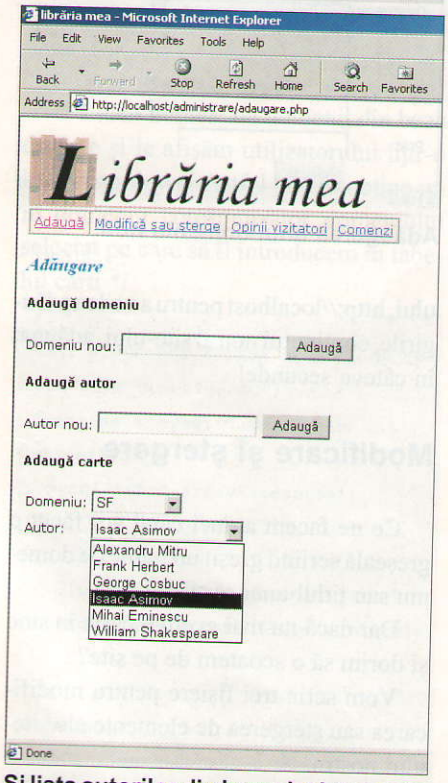
adaugare.php

```
<?
include("autorizare.php");
include("admin_top.php");
?>
<h1>Adăugare</h1>
<b>Adaugă domeniu</b>
 <form action="prelucrare_adaugare.php"
 method="POST">
    Domeniu nou: <INPUT type="text"
    name="domeniu_nou">
    <INPUT type="submit"
    name="adauga_domeniu"
    value="Adaugă">
 </form>
<b>Adaugă autor</b>
 <form
 action="prelucrare_adaugare.php"
 method="POST">
    Autor nou: <INPUT type="text"
    name="autor_nou">
    <INPUT type="submit"
    name="adauga_autor"
    value="Adaugă">
 </form>
<b>Adaugă carte</b>
<form
action="prelucrare_adaugare.php"
method="POST">
<table>
<tr>
 <td>Domeniu:</td>
 <td>
  <select name="id_domeniu">
/* Luăm numele de domenii din baza de date şi le afişăm utlizatorului într-o listă drop-down. Astfel putem obţine un id_domeniu corespunzător domeniului selectat pe care să îl introducem în tabelul carti */
```



Alegerea unui domeniu dintr-o listă generată automat.

```
<?
$sql = "SELECT * FROM domenii
 ORDER BY nume_domeniu ASC";
$resursa = mysql_query($sql);
while($row=mysql_fetch_array($resursa))
{
 print '<option value=
 "'.$row['id_domeniu'].'">'.
 $row['nume_domeniu'].'</option>';
}
?>
</select>
</td>
</tr>
<tr>
<td>Autor:</td>
<td>
 <select name="id_autor">
```



Şi lista autorilor din baza de date poate fi generată automat.

/* Afişăm şi lista dropdown cu autori */

```
<?
$sql = "SELECT * FROM autori ORDER
 BY nume_autor ASC";
$resursa = mysql_query($sql);
while($row =
 mysql_fetch_array($resursa))
{
 print '<option
 value="'.$row['id_autor'].'">'.
 $row['nume_autor'].'</option>';
}
?>
</select>
</td>
</tr>
<tr>
 <td>Titlu:</td>
 <td><INPUT type="text" name="titlu">
 </td>
</tr>
<tr>
 <td valign="top">Descriere:</td>
 <td><textarea name="descriere"
 rows="8"></textarea></td>
</tr>
<tr>
 <td>Pret:</td>
 <td><INPUT type="text" name="pret">
 </td>
</tr>
<tr>
 <td></td>
 <td><INPUT type="submit" name=
 "adauga_carte" value="Adaugă"></td>
</tr>
</table>
</form>
</body>
</html>
```

prelucrare_adaugare.php

```
<?
include("autorizare.php");
include("admin_top.php");
```

/* Dacă s-a trimis formularul pentru adăugare de domeniu */

```
if(isset($_POST['adauga_domeniu']))
{
```

/* Înainte de a introduce noul nume de domeniu verificăm două lucruri: să nu fie gol şi să nu existe deja în baza de date. În cazul în care numele de domeniu este gol sau se află deja în tabelul domenii, afişăm un mesaj de eroare şi oprim execuţia scriptului: */

```
if($_POST['domeniu_nou'] == "")
{
 print 'Trebuie să completezi
  numele de domeniu!<br>
  <a href="adaugare.php">Înapoi</a>';
 exit;
}
```

/* Verificăm dacă nu există deja în baza de date: */

```
$sql = "SELECT * FROM domenii WHERE
nume_domeniu='".$_POST['domeniu_nou']."'";
$resursa = mysql_query($sql);
```

/* Interogarea returnează 0 rânduri dacă domeniul nu există în baza de date. Dacă nu returnează 0 înseamnă că domeniul specificat există deja în baza de date. Nu îl vom adăuga din nou ci doar vom atenţiona utilizatorul de eroare şi vom întrerupe execuţia ulterioară a scriptului:*/

```
if(mysql_num_rows($resursa) != 0)
{
 print 'Domeniul
  <b>'.$_POST['domeniu_nou'].'</b>
  există deja în baza de date!<br>
  <a href="adaugare.php">Înapoi</a>';
 exit;
}
```

/* Am verificat să nu fie erori, putem acum să adăugăm noul nume de domeniu în baza de date: */

```
$sql = "INSERT INTO domenii
 (nume_domeniu) VALUES
 ('".$_POST['domeniu_nou']."')";
mysql_query($sql);
```

/* şi afişăm utilizatorului un mesaj de confirmare: */

```
print 'Domeniul
 <b>'.$_POST['domeniu_nou'].'</b> a
 fost adăugat în baza de date!<br>
 <a href="adaugare.php">Înapoi</
 a>';
exit;
}
```

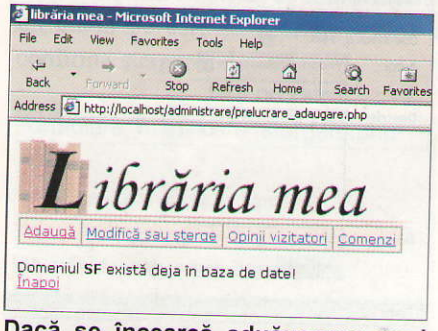
/* Acelaşi script, cu mici diferenţe, îl vom folosi pentru adăugarea unui autor nou, dacă s-a trimis formularul de adăugare autor: */

```
if(isset($_POST['adauga_autor']))
{
```

/* Verificăm dacă nu este gol: */

```
if($_POST['autor_nou'] == "")
{
 print 'Trebuie să completezi numele
  autorului!<br>
  <a href="adaugare.php">Înapoi</a>';
 exit;
}
```

/* Verificăm dacă nu există deja în baza de date: */



Dacă se încearcă adăugarea unui domeniu deja existent primim acest mesaj

```
$sql = "SELECT * FROM autori WHERE
  nume_autor='".$_POST['autor_nou']."'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) != 0)
{
  print 'Autorul
    <b>'.$_POST['autor_nou'].'</b>
    există deja în baza de date!<br>
    <a href="adaugare.php">Înapoi</a>';
  exit;
}
```

/* Am verificat să nu fie erori, putem acum să adăugăm noul autor în baza de date: */

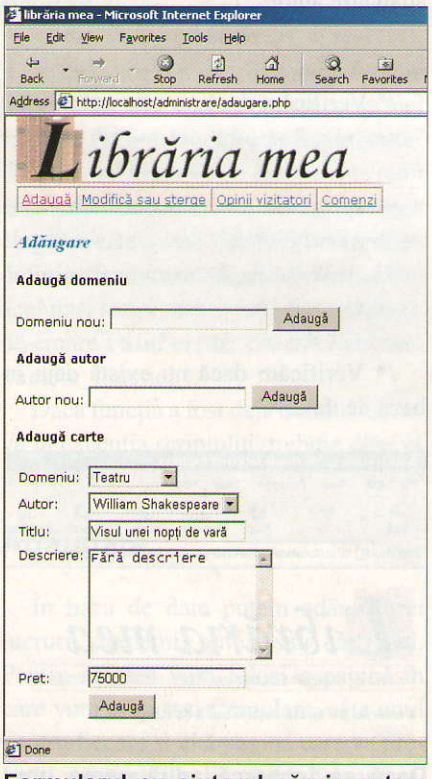
```
$sql = "INSERT INTO autori(nume_autor)
  VALUES('".$_POST['autor_nou']."')";
mysql_query($sql);
```

/* şi afişăm utilizatorului un mesaj de confirmare: */

```
print 'Autorul
  <b>'.$_POST['autor_nou'].'</b> a
  fost adăugat în baza de date!<br>
  <a href="adaugare.php">Înapoi</a>';
exit;
}
```

/* Scriptul pentru adăugarea cărţii va fi o idee mai complicat deoarece avem mai multe variabile de verificat, dar în mare are acelaşi mod de funcţionare: */

```
if(isset($_POST['adauga_carte']))
{
```



Formularul cu ajutorul căruia putem adăuga rapid o carte.

/* Verificăm dacă titlul, descrierea sau preţul nu sunt goale: */

```
if($_POST['titlu'] == "" ||
$_POST['descriere'] == "" ||
$_POST['pret'] == "")
{
  print 'Titlul, descrierea sau
    preţul sunt goale!<br>
    <a href="adaugare.php">Înapoi</a>';
  exit;
}
```

/* Verificăm dacă valoarea introdusă în câmpul Preţ este de tip numeric folosind funcţia is_numeric: */

```
if(!is_numeric($_POST['pret']))
{
  print 'Câmpul Preţ trebuie să fie
    de tip numeric! (scrieţi <b>1000
    </b> în loc de <b>1000 lei</b>)<br>
    <a href="adaugare.php">Înapoi</a>';
  exit;
}
```

/* Verificăm dacă această carte nu există deja în baza de date, după două criterii: id_autor şi titlu. */

```
$sql = "SELECT * FROM carti WHERE
  id_autor='".$_POST['id_autor']."'
  AND titlu='".$_POST['titlu']."'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) != 0)
{
  print 'Această carte există deja în
    baza de date!<br>
    <a href="adaugare.php">Înapoi</a>';
  exit;
}
```

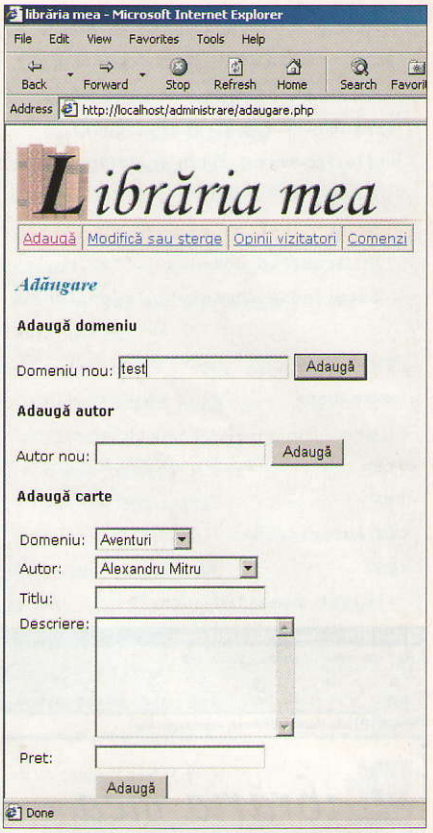
/* Am verificat să nu fie erori, putem acum să adăugăm cartea în baza de date.*/

```
$sql = "INSERT INTO carti(id_domeniu,
  id_autor, titlu, descriere, pret)
  VALUES('".$_POST['id_domeniu']."','".
  $_POST['id_autor']."','".$_POST['titlu']."','".
  $_POST['descriere']."','".$_POST['pret']."')";
  mysql_query($sql);
```

/* Şi afişăm utilizatorului un mesaj de confirmare: */

```
print 'Cartea a fost adăugată în baza
  de date!<br>
  <a href="adaugare.php">Înapoi</a>';
exit;
}
?>
</body>
</html>
```

Rulaţi aceste scripturi pentru a introduce cărţi, autori şi domenii noi în baza de date. Accesaţi prima pagină a site-



Adăugarea unui domeniu nou.

ului, http://localhost pentru a vedea adăugirile, conţinutul nou al site-ului, adăugat în câteva secunde!

Modificare şi ştergere

Ce ne facem atunci când am făcut o greşeală scriind greşit un nume de domeniu sau titlul unei cărţi?

Dar dacă nu mai avem o carte în stoc şi dorim să o scoatem de pe site?

Vom scrie trei fişiere pentru modificarea sau ştergerea de elemente ale site-ului nostru.

Fişierul modificare_stergere.php va conţine trei formulare: unul pentru numele de domenii, unul pentru numele autorilor şi altul pentru cărţi.

Numele de domenii şi numele autorilor le vom afişa în liste dropdown, astfel încât să putem selecta imediat domeniul sau autorul ce dorim să îl ştergem.

Formularul pentru cărţi va conţine două câmpuri: autor şi titlu. Numele autorului va fi disponibil într-o listă dropdown, iar titlul cărţii îl vom scrie într-un <input type="text">.

Iată codul acestui script şi puteţi vedea în imaginea de la pag. 58 rezultatul său:

modificare_stergere.php

```
<?
include("autorizare.php");
include("admin_top.php");
?>
<h1>Modificare sau ştergere</h1>
  <p><b>Notă:</b> Nu veţi putea
  şterge domenii care au cărţi în
  ele. Înainte de a şterge domeniul,
  modificaţi cărţile din el astfel
  încât să aparţină altor domenii.
  De asemenea nu veţi putea şterge
  un autor dacă există cărţi în baza
  de date care au acel autor.</p>
  <b>Selectează domeniul ce doreşti
  să îl modifici sau ştergi:</b>
<form
action="formulare_modificare_stergere.php"
 method="POST">
Domeniu:
 <select name="id_domeniu">
```

/* Luăm numele de domenii din baza de date şi le afişăm utilizatorului într-o listă drop-down. Astfel putem obţine un id_domeniu corespunzător domeniului selectat pe care să îl introducem în tabelul carti */

```
<?
$sql = "SELECT * FROM domenii ORDER
  BY nume_domeniu ASC";
$resursa = mysql_query($sql);
while($row =
  mysql_fetch_array($resursa))
{
 print '<option
  value="'.$row['id_domeniu'].'">'
  .$row['nume_domeniu'].'</option>';
}
?>
 </select>
<INPUT type="submit" name=
"modifica_domeniu" value="Modifică">
<INPUT type="submit" name=
"sterge_domeniu" value="Şterge">
</form>

<b>Selectează autorul ce doreşti să
îl modifici sau ştergi:</b>
<form
action="formulare_modificare_stergere.php"
method="POST">
Autor:
 <select name="id_autor">
```

/* Afişăm şi lista dropdown cu autori */

```
<?
$sql = "SELECT * FROM autori ORDER BY
  nume_autor ASC";
$resursa = mysql_query($sql);
while($row =
  mysql_fetch_array($resursa))
{
 print '<option
  value="'.$row['id_autor'].'">'
  .$row['nume_autor'].'</option>';
}
?>
 </select>
<INPUT type="submit" name=
"modifica_autor" value="Modifică">
<INPUT type="submit"
name="sterge_autor" value="Şterge">
</form>
<b>Selectează autorul şi scrie tit-
lul cărţii ce doreşti să o modifici
sau ştergi:</b>
<form action="formulare_modificare_
stergere.php" method="POST">
<table>
 <tr>
 <td>Autor:</td>
 <td>
  <select name="id_autor">
  <?
```

/* Afişăm şi lista dropdown cu autori */

```
  $sql = "SELECT * FROM autori ORDER
    BY nume_autor ASC";
  $resursa = mysql_query($sql);
  while($row =
    mysql_fetch_array($resursa))
  {
   print '<option
    value="'.$row['id_autor'].'">'
    .$row['nume_autor'].'</option>';
  }
  ?>
  </select>
 </td>
 </tr>
 <tr>
 <td>Titlu:</td>
 <td><INPUT type="text" name="titlu">
 </td>
 </tr>
</table>
 <INPUT type="submit" name=
  "modifica_carte" value="Modifică">
 <INPUT type="submit" name=
  "sterge_carte" value="Şterge">
</form>
</body>
</html>
```

Fişierul formulare_modificare_stergere.php va prelua informaţiile din modificare_stergere.php şi va afişa câte un formular pentru fiecare caz în parte:

- dacă a fost apăsat butonul de modificare nume domeniu, va afişa numele de domeniu într-un textbox astfel încât să îl putem modifica;
- dacă a fost apăsat butonul de ştergere domeniu, va verifica întâi dacă sunt cărţi în domeniul respectiv. Dacă sunt, va afişa lista acestora şi va afişa mesajul „Acest domeniu nu poate fi şters deoarece sunt cărţi care îi aparţin!". Dacă nu este nici o carte în domeniul respectiv, va afişa un mesaj cu o cerere de confirmare şi un buton pe care să apăsăm pentru a confirma ştergerea.
- similar, dacă a fost apăsat butonul de modificare nume autor, va afişa numele autorului într-un textbox astfel încât să îl putem modifica;
- dacă a fost apăsat butonul de ştergere autor, va verifica întâi dacă sunt cărţi în baza de date care corespund autorului. Dacă sunt, va afişa lista acestora şi va afişa mesajul „Acest autor nu poate fi şters deoarece sunt cărţi în baza de date care îi aparţin!". Dacă nu este nici o carte scrisă de autor în baza de date, va afişa un mesaj cu o cerere de confirmare şi un buton pe care să apăsăm pentru a confirma ştergerea.
- atunci când se apasă butonul de modificare carte, verifică întâi dacă există o carte în baza de date care are titlul şi autorul menţionate (titlul este scris de către utilizator şi trebuie să ne asigurăm că l-a scris corect). Dacă nu există nici o carte cu datele menţionate afişează un mesaj de eroare. Dacă în schimb cartea există, afişează un formular cu ajutorul căruia să putem modifica domeniul, autorul, titlul, descrierea şi preţul cărţii.
- dacă s-a apăsat butonul de ştergere carte, verifică dacă există o carte cu titlul şi autorul menţionate. Dacă nu există afişează un mesaj de eroare iar dacă există afişează un mesaj de confirmare şi un buton pe care să apăsăm pentru a confirma ştergerea.

formulare_modificare_stergere.php

```
<?
include("autorizare.php");
include("admin_top.php");
/* modificare nume domeniu */
if(isset($_POST['modifica_domeniu']))
{
```

/* luăm numele de domeniu din baza

de date deoarece ne-a fost trimis din formular doar id-ul domeniului: */

```
$sql = "SELECT nume_domeniu FROM
 domenii WHERE
 id_domeniu='".$_POST['id_domeniu']."'";
 $resursa = mysql_query($sql);
 $nume_domeniu =
 mysql_result($resursa, 0,
 "nume_domeniu");
```

/* şi afişăm numele vechi de domeniu într-un textbox pentru a fi modificat */

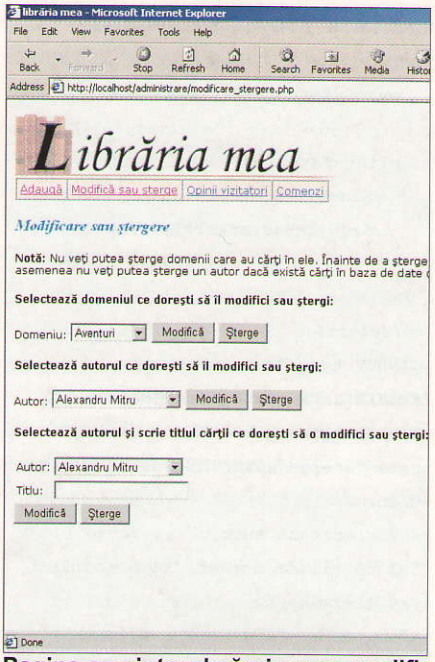
```
 ?>
 <h1>Modifică nume domeniu</h1>
 <form action=
 "prelucrare_modificare_stergere.php"
 method="POST">
  <INPUT type="text"
   name="nume_domeniu"
   value="<?=$nume_domeniu?>">
  <INPUT type="hidden"
   name="id_domeniu"
   value="<?=$_POST['id_domeniu']?>">
  <INPUT type="submit"
   name="modifica_domeniu"
   value="Modifică">
 </form>
 <?
}
/* ştergere domeniu */
if(isset($_POST['sterge_domeniu']))
{
```

/* verificăm dacă sunt cărţi în baza de date care aparţin acestui domeniu: */

```
 $sql = "SELECT titlu, nume_autor
  FROM carti, autori, domenii WHERE
  carti.id_domeniu=domenii.id_domeniu
  AND carti.id_autor=autori.id_autor
  AND domenii.id_domeniu=
  ".$_POST['id_domeniu'];
 $resursa = mysql_query($sql);
 $nrCarti = mysql_num_rows($resursa);
```

/* dacă sunt cărţi aparţinând acestui domeniu, afişăm lista lor şi un mesaj de eroare: */

```
 if($nrCarti > 0)
 {
  print "<p>Sunt $nrCarti cărţi
   care aparţin acestui domeniu!</p>";
  while($row =
   mysql_fetch_array($resursa))
  {
   print "<b>".$row['titlu']."</b>
    de ".$row['nume_autor']."<br>";
  }
  print "<p>Nu puteţi şterge acest
   domeniu!</p>";
 }
```



Pagina cu ajutorul căreia vom modifica sau şterge informaţii de pe site.

/* iar dacă nu sunt cărţi în acest domeniu cerem confirmarea pentru ştergere: */

```
 else
 {
  ?>
  <h1>Şterge nume domeniu</h1>
  Eşti sigur că vrei să ştergi ac-
  est domeniu?
  <form
  action="prelucrare_modificare_stergere.php"
  method="POST">
  <INPUT type="hidden" name="id_domeniu"
   value="<?=$_POST['id_domeniu']?>">
  <INPUT type="submit" name=
   "sterge_domeniu" value="Şterge!">
  </form>
  <?
 }
}
```

/* modificare nume autor */

```
if(isset($_POST['modifica_autor']))
{
```

/* luăm numele autorului din baza de date deoarece ne-a fost trimis din formular doar id_autor: */

```
 $sql = "SELECT nume_autor FROM
  autori WHERE id_autor=
  '".$_POST['id_autor']."'";
 $resursa = mysql_query($sql);
 $nume_autor = mysql_result
  ($resursa, 0, "nume_autor");
```

/* şi afişăm numele într-un textbox pentru a fi modificat */

```
 ?>
 <h1>Modifică nume autor</h1>
 <form
 action="prelucrare_modificare_stergere.php"
 method="POST">
 <INPUT type="text"
  name="nume_autor"
  value="<?=$nume_autor?>">
 <INPUT type="hidden"
  name="id_autor"
  value="<?=$_POST['id_autor']?>">
 <INPUT type="submit"
  name="modifica_autor"
  value="Modifică">
 </form>
 <?
}
/* ştergere autor */
if(isset($_POST['sterge_autor']))
{
```

/* verificăm dacă sunt cărţi în baza de date care aparţin acestui autor: */

```
 $sql = "SELECT titlu FROM carti,
  autori WHERE carti.id_autor=
  autori.id_autor AND
  carti.id_autor=
  ".$_POST['id_autor'];
 $resursa = mysql_query($sql);
 $nrCarti = mysql_num_rows($resursa);
```

/* dacă sunt cărţi aparţinând acestui autor, afişăm lista lor şi un mesaj de eroare: */

```
 if($nrCarti > 0)
 {
  print "<p>Sunt $nrCarti cărţi de
   acest autor în baza de date!</p>";
  while($row =
   mysql_fetch_array($resursa))
  {
   print $row['titlu']."<br>";
  }
  print "<p>Nu puteţi şterge acest
   autor!</p>";
 }
```

/* iar dacă nu sunt cărţi de acest autor cerem confirmarea pentru ştergere: */

```
 else
 {
  ?>
  <h1>Şterge autor</h1>
  Eşti sigur că vrei să ştergi
  acest autor?
  <form
  action="prelucrare_modificare_stergere.php"
  method="POST">
  <INPUT type="hidden"
   name="id_autor"
   value="<?=$_POST['id_autor']?>">
```

```
  <INPUT type="submit"
   name="sterge_autor"
   value="$terge!">
  </form>
  <?
 }
}
/* modificare carte */
if(isset($_POST['modifica_carte']))
{
 print "<h1>Modificare carte</h1>";
```

/* căutăm întâi o carte în baza de date care are titlul şi id_autor specificate în formular: */

```
$sqlCarte = "SELECT * FROM carti
 WHERE titlu='".$_POST['titlu']."'
 AND id_autor=".$_POST['id_autor'];
$resursaCarte = mysql_query($sqlCarte);
```

/* dacă nu s-a găsit nici o carte care să corespundă datelor introduse, afişăm un mesaj de eroare: */

```
if(mysql_num_rows($resursaCarte) == 0)
{
 print "Această carte nu există în
  baza de date";
}
else
{
```

/* dacă există, atunci extragem informaţiile din resursă, le punem într-un array (nu folosim while deoarece este returnat un singur rând!) şi le afişăm în formular pentru a fi modificate: */

```
   $rowCarte =
    mysql_fetch_array($resursaCarte);
   ?>
   <form
   action="prelucrare_modificare_stergere.php"
   method="POST">
   <table>
   <tr>
   <td>Domeniu:</td>
   <td>
   <select name="id_domeniu">
   <?
```

/* Luăm numele de domenii din baza de date şi le afişăm utlizatorului într-o listă drop-down. Observaţi folosirea lui if pentru a afişa ca selectat domeniul de care aparţine cartea: */

```
   $sql = "SELECT * FROM domenii
    ORDER BY nume_domeniu ASC";
   $resursa = mysql_query($sql);
   while($row =
    mysql_fetch_array($resursa))
   {
```

```
    if($row['id_domeniu'] ==
     $rowCarte['id_domeniu'])
    {
     print '<option SELECTED
      value="'.$row['id_domeniu'].'">'.
      $row['nume_domeniu'].'
      </option>';
    }
    else
    {
     print '<option
      value="'.$row['id_domeniu'].'">'.
      $row['nume_domeniu'].'
      </option>';
    }
   }
   ?>
   </select>
   </td>
   </tr>
   <tr>
   <td>Autor:</td>
   <td>
   <select name="id_autor">
   <?
```

/* Afişăm şi lista dropdown cu autori */

```
$sql = "SELECT * FROM autori
 ORDER BY nume_autor ASC";
$resursa = mysql_query($sql);
while($row =
 mysql_fetch_array($resursa))
{
 if($row['id_autor'] ==
  $rowCarte['id_autor'])
 {
  print '<option SELECTED
   value="'.$row['id_autor'].'">'.
   $row['nume_autor'].'
   </option>';
 }
 else
 {
  print '<option
   value="'.$row['id_autor'].'">'.
   $row['nume_autor'].'
   </option>';
 }
}
?>
</select>
</td>
</tr>
<tr>
<td>Titlu:</td>
<td>
<INPUT type="text" name="titlu"
```

```
    value="<?=$rowCarte['titlu']?>">
   </td>
   </tr>
   <tr>
   <td valign="top">Descriere:</td>
   <td><textarea name="descriere"
   rows="8"><?=$rowCarte['descriere']?>
   </textarea></td>
   </tr>
   <tr>
   <td>Pret:</td>
   <td><INPUT type="text" name="pret"
    value="<?=$rowCarte['pret']?>">
   </td></tr>
   </table>
   <INPUT type="hidden"
    name="id_carte"
    value="<?=$rowCarte['id_carte']?>">
   <INPUT type="submit"
    name="modifica_carte"
    value="Modifică">
   </form>
   <?
 }
}
```

/* şi în final ştergere carte */

```
if(isset($_POST['sterge_carte']))
{
 print "<h1>Şterge carte</h1>";
```

/* căutăm întâi o carte în baza de date care are titlul şi id_autor specificate în formular: */

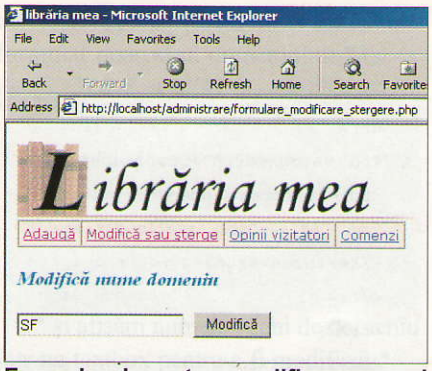
```
 $sqlCarte = "SELECT * FROM carti
  WHERE titlu='".$_POST['titlu']."'
  AND id_autor=".$_POST['id_autor'];
 $resursaCarte=mysql_query($sqlCarte);
```

/* dacă nu s-a găsit nici o carte care să corespundă datelor introduse, afişăm un mesaj de eroare: */

```
 if(mysql_num_rows($resursaCarte)==0)
 {
  print "Această carte nu există în
   baza de date";
 }
```

/* iar dacă există, atunci extragem id-ul cărţii din baza de date şi îl vom folosi într-un câmp ascuns din formularul de confirmare: */

```
 else
 {
  $id_carte = mysql_result
   ($resursaCarte,0,id_carte");
  ?>
  Eşti sigur că vrei să ştergi
  această carte?
  <form
  action="prelucrare_modificare_stergere.php"
```



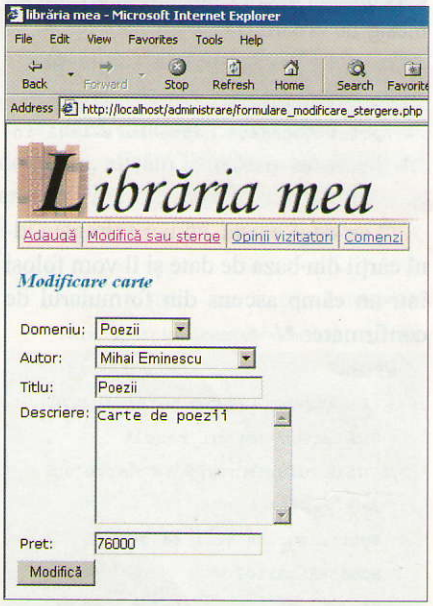
Formularul pentru modificarea unui nume de domeniu scris greşit.

```
    method="POST">
    <INPUT type="hidden"
     name="id_carte"
     value="<?=$id_carte?>">
    <INPUT type="submit"
     name="sterge_carte"
     value="Șterge!">
    </form>
    <?
  }
}
?>
</body>
</html>
```

După cum vedem, pagina formulare_modificare_stergere.php conține doar formulare, pentru toate operațiunile de modificare şi ştergere din baza de date. Să mai facem un ultim fişier, în care să prelucrăm datele trimise prin aceste formulare. Structura sa va fi următoarea:

- modificarea numelui de domeniu în baza de date;
- (sau) ştergerea domeniului din baza de date;
- (sau) modificarea numelui autorului ;
- (sau) ştergerea autorului;
- (sau) actualizarea informațiilor despre carte;
- (sau) ştergerea cărții din baza de date şi a tuturor comentariilor utilizatorilor la cartea respectivă.

Codul este următorul (semnificativ mai simplu decât cel al fişierului precedent):



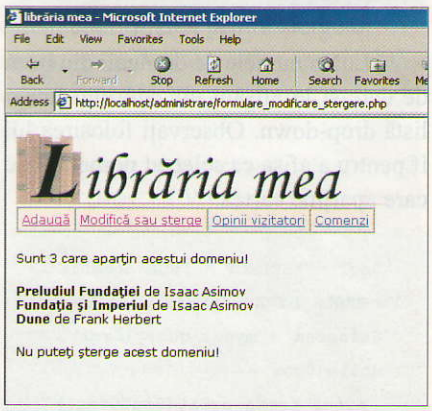
Modificarea informațiilor unei înregistrări este simplă.

prelucrare_modificare_stergere.php

```
<?
include("autorizare.php");
include("admin_top.php");
/* modificare nume domeniu */
if(isset($_POST['modifica_domeniu']))
{
 /* Verificăm dacă noul nume de domeniu a fost introdus. */
 if($_POST['nume_domeniu'] == "")
 {
  print "Nu ați introdus numele
   domeniului!";
 }
 else
 {
  $sql = "UPDATE domenii SET
   nume_domeniu='".$_POST['nume_domeniu']."'
   WHERE id_domeniu=".
   $_POST['id_domeniu'];
  mysql_query($sql);
  print "Numele domeniului a fost
   modificat!";
 }
 /* De ce nu am folosi exit în structura if(condiție){codul de executat; exit;}? Pentru că dacă nu se execută codul din if(conditie){}, se execută codul din else{} şi atunci exit ar fi superfluu. */
}
 /* ştergere domeniu */
if(isset($_POST['sterge_domeniu']))
{
 $sql = "DELETE FROM domenii WHERE
  id_domeniu=".$_POST['id_domeniu'];
 mysql_query($sql);
 print "Domeniul a fost şters!";
}
 /* modificare nume autor */
if(isset($_POST['modifica_autor']))
{
 /* Verificăm dacă numele modificat al autorului a fost introdus. */
 if($_POST['nume_autor'] == "")
 {
  print "Nu ați introdus numele autorului!";
 }
 else
 {
  $sql="UPDATE autori SET nume_autor=
   '".$_POST['nume_autor']."' WHERE
   id_autor=".$_POST['id_autor'];
  mysql_query($sql);
  print "Numele autorului a fost
   modificat!";
 }
}
/* ştergere autor */
if(isset($_POST['sterge_autor']))
{
 $sql = "DELETE FROM autori WHERE
  id_autor=".$_POST['id_autor'];
 mysql_query($sql);
 print "Autorul a fost şters!";
}
/* modificare informații carte */
if(isset($_POST['modifica_carte']))
{
```

/* Verificăm dacă toate datele au fost introduse corect. N-am vrea să introducem date eronate în baza de date doar pentru că a sărit pisica pe tastatură şi a apăsat ENTER în timp ce introduceam datele. Dacă credeți că nu vi se poate întâmpla... ei bine, din proprie experiență vă spun că se poate. Vom folosi o structură if ... else if... else: */

```
 if($_POST['titlu'] == "")
 {
  print "Nu ați introdus titlul!";
 }
  else if($_POST['descriere'] == "")
 {
  print "Nu ați introdus descrierea!";
 }
```



Nu putem şterge un domeniu în care există cărți.

```
    else if($_POST['pret'] == "")
{
    print "Nu ați introdus prețul!";
}
    else if(!is_numeric($_POST['pret']))
{
    print "Prețul trebuie să fie
     numeric! Scrieți <b>1000</b>,
     nu <b>1000 lei</b>!";
}
else
{
    $sql = "UPDATE carti SET
     id_domeniu=".$_POST['id_domeniu'].",
     id_autor=".$_POST['id_autor'].",
     titlu='".$_POST['titlu']."',
     descriere='".$_POST['descriere']."',
     pret=".$_POST['pret']." WHERE
     id_carte=".$_POST['id_carte'];
    mysql_query($sql);
    print "Informațiile au fost
    modificate!";
}
}
/* ștergere carte */
if(isset($_POST['sterge_carte']))
{
 $sqlCarte = "DELETE FROM carti
  WHERE id_carte="
  .$_POST['id_carte'];
 mysql_query($sqlCarte);
 $sqlComentarii="DELETE FROM comentarii
  WHERE id_carte=".$_POST['id_carte'];
 mysql_query($sqlComentarii);
 print "Cartea a fost ștearsă din
  baza de date!";
}
?>
</body>
</html>
```

Moderare comentarii

Următorul lucru de care ne vom ocupa va fi moderarea comentariilor vizitatorilor. Deşi utilizatorul obişnuit este bine crescut şi cunoaşte regulile netichetei nu toţi sunt aşa şi există posibilitatea să trebuiască să editaţi sau chiar să ştergeţi unele comentarii.

Să monitorizăm fiecare pagină de carte în fiecare zi pentru a vedea ce comentarii noi au fost postate nu este o soluţie viabilă: necesită mult timp şi ar fi practic imposibilă în cazul unui site cu multe cărţi. De aceea, vom avea în zona de administrare o secţiune specială pentru moderare unde vom afişa comentariile noi (adăugate de la ultima moderare).

Pentru a afla care sunt comentariile noi de la ultima accesare a secţiunii de moderare va trebui să specificăm undeva care este ultimul comentariu gata moderat. Nu vom seta un cookie pentru că atunci am fi condiţionaţi de folosirea aceluiaşi calculator şi browser pentru accesarea paginii de administrare. Vom adăuga încă un câmp în tabelul admin unde vom păstra id-ul ultimului comentariu moderat.

Cum id-urile sunt create în ordine crescătoare, comentariile noi vor fi cele care au id mai mare decât cel setat în tabelul admin. Să creăm noul câmp:

```
ALTER TABLE admin ADD
 ultimul_comentariu_moderat INT
 UNSIGNED DEFAULT "0" NOT NULL;
```

Ultimul comentariu moderat este în momentul de faţă 0. Comentariile noi le putem afla cu interogarea:

```
SELECT * FROM comentarii, admin
 WHERE comentarii.id_comentariu >
 admin.ultimul_comentariu_moderat;
```

Pentru a afla şi titlurile şi autorii cărţilor la care sunt comentarii noi, interogarea este:

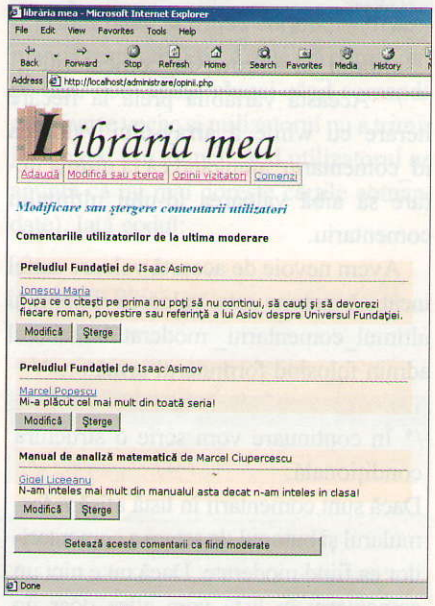
```
SELECT * FROM comentarii, admin,
 carti, autori WHERE id_comentariu >
 admin.ultimul_comentariu_moderat AND
 carti.id_carte=comentarii.id_carte
 AND carti.id_autor=autori.id_autor;
```

Vom folosi această interogare în pagina de moderare comentarii şi vom adăuga un formular cu ajutorul căruia să putem modifica valoarea câmpului ultimul_comentariu_moderat la fiecare vizită astfel încât dacă azi moderăm comentariile, mâine să nu vedem decât comentariile adăugare după ultimul comentariu moderat de azi. De asemenea, vom avea butoane pentru modificare sau ştergere comentariu.

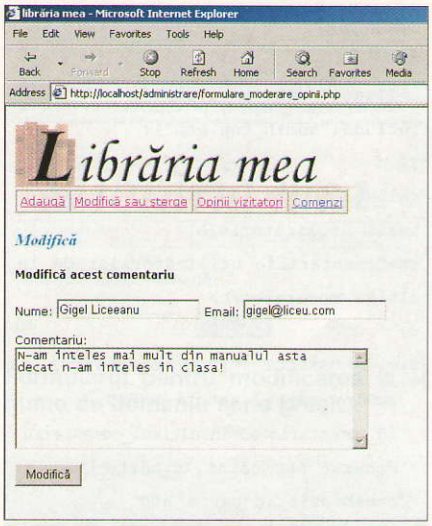
După ce vom revizui comentariile utilizatorilor, modificându-le sau ştergându-le pe cele nepotrivite, vom putea apăsa butonul „Setează aceste comentarii ca fiind moderate" pentru a nu le mai vedea la următoarea verificare.

opinii.php

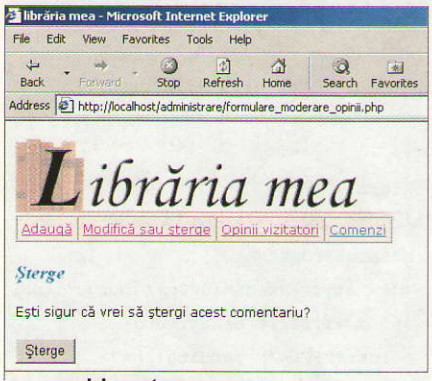
```
<?
include("autorizare.php");
include("admin_top.php");
?>
<h1>Modificare sau ștergere comen-
tarii utilizatori</h1>
<b>Comentariile utilizatorilor de la
ultima moderare</b>
<?
$sql = "SELECT * FROM comentarii,
  admin, carti, autori WHERE
  id_comentariu>admin.ultimul_comentariu_
  moderat AND carti.id_carte=
  comentarii.id_carte AND
  carti.id_autor=autori.id_autor
  ORDER BY id_comentariu ASC";
$resursa = mysql_query($sql);
while($row =
  mysql_fetch_array($resursa))
{
 ?>
 <form
 action="formulare_moderare_opinii.php"
 method="POST">
 <div style="width:500px; border:1px
 solid #ffffff; background-
 color:#F9F1E7; padding:5px">
  <b><?=$row['titlu']?></b> de
  <?=$row['nume_autor']?>
  <HR size="1">
  <a href=
   "mailto:<?=$row['adresa_email']?>">
   <?=$row['nume_utilizator']?>
   </a><br>
  <?=$row['comentariu']?>
</div>
```



Comentariile noi şi încă nemoderate ale utilizatorilor site-ului.



Formularul cu ajutorul căruia putem edita un comentariu nepotrivit...



...sau chiar şterge.

```
<INPUT type="hidden"
  name="id_comentariu"
  value="<?=$row['id_comentariu']?>">
<INPUT type="submit"
   name="modifica" value="Modifică">
<INPUT type="submit" name="sterge"
   value="Şterge">
</form>
<?
```

$ultimul_id = $row['id_comentariu'];

/* Această variabilă preia la fiecare iterare cu while a array-ului valoarea id_comentariu, ajungând ca la ultima iterare să aibă valoarea id-ului ultimului comentariu.

Avem nevoie de această valoare astfel încât să putem seta valoarea câmpului ultimul_comentariu_ moderat din tabelul admin folosind formularul următor. */

```
}
```

/* În continuare vom scrie o structură condiţională.

Dacă sunt comentarii în listă afişăm formularul şi butonul de setare a comentariilor ca fiind moderate. Dacă nu e nici un comentariu în listă vom afişa doar un mesaj. Astfel evităm erorile care ar putea apărea dacă nu avem comentarii în listă şi valoarea variabilei $ultimul_comentariu ar fi nulă. */

```
$nrComentarii=mysql_num_rows($resursa);
if($nrComentarii > 0)
{
  ?>
  <form
  action="formulare_moderare_opinii.php"
  method="POST">
  <INPUT type="hidden"
    name="ultimul_id"
    value="<?=$ultimul_id?>">
  <INPUT type="submit"
    name="seteaza_moderate"
    value="Setează aceste comentarii
    ca fiind moderate">
  </form>
  <?
}
else
{
  print "<p>Nu există comentarii
    noi.</p>";
}
?>
</body>
</html>
```

Fişierul care preia informaţiile din opinii.php va afişa 3 formulare, în funcţie de butonul apăsat.

- dacă s-a apăsat butonul „Modifică" afişează un formular astfel încât să putem edita conţinutul mesajului;
- dacă s-a apăsat butonul „Şterge" afişează un mesaj de confirmare şi un buton pe care putem apăsa pentru a confirma ştergerea;
- dacă s-a apăsat butonul „Setează aceste comentarii ca fiind moderate", afişează un mesaj şi un buton pe care putem apăsa pentru a confirma că am terminat moderarea comentariilor din pagina precedentă.

formulare_moderare_opinii.php

```
<?
include("autorizare.php");
include("admin_top.php");
/* formular modificare comentariu*/
if(isset($_POST['modifica']))
{
 $sql = "SELECT * FROM comentarii WHERE
   id_comentariu=".$_POST['id_comentariu'];
 $resursa = mysql_query($sql);
```

/* fiind returnat un singur rând, nu folosim while */

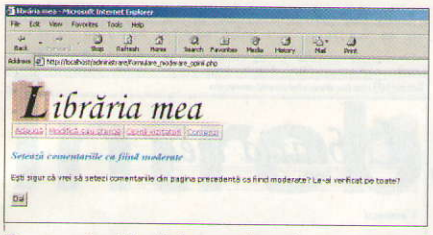
```
 $row = mysql_fetch_array($resursa);
 ?>
 <h1>Modifică</h1>
 <b>Modifică acest comentariu</b>
 <form
 action="prelucrare_moderare_comentarii.php"
 method="POST">
   Nume: <input type="text"
     name="nume_utilizator"
     value="<?=$row['nume_utilizator']?>">
   Email: <input type="test"
     name="adresa_email"
     value="<?=$row['adresa_email']?>"><br><br>
   Comentariu: <br><textarea name=
     "comentariu" cols="45" rows="8">
     <?=$row['comentariu']?>
     </textarea><br><br>
   <input type="hidden"
     name="id_comentariu"
     value="<?=$_POST['id_comentariu']?>">
   <input type="submit"
     name="modifica" value="Modifică">
 </form>
 <?
}
```

/* confirmare ştergere comentariu */

```
if(isset($_POST['sterge']))
{
  ?>
  <h1>Şterge</h1>
  Eşti sigur că vrei să ştergi acest
  comentariu?
  <form
  action="prelucrare_moderare_comentarii.php"
  method="POST">
    <input type="hidden"
      name="id_comentariu"
      value="<?=$_POST['id_comentariu']?>">
    <input type="submit"
      name="sterge" value="Şterge">
  </form>
  <?
}
```

/* confirmare moderare*/

```
if(isset($_POST['seteaza_moderate']))
{
  ?>
  <h1>Setează comentariile ca fiind
  moderate</h1>
  Eşti sigur că vrei să setezi comen-
  tariile din pagina precedentă ca fiind
  moderate? Le-ai verificat pe toate?
  <form
  action="prelucrare_moderare_comentarii.php"
  method="POST">
```



Comentariile deja moderate nu vor mai apărea la următoarea vizită în secțiunea administrativă.

Nu există nici un comentariu nou nemoderat de la ultima accesare a secțiunii de administrare.

```
    <input type="hidden"
      name="ultimul_id"
      value="<?=$_POST['ultimul_id']?>">
    <input type="submit" name="
      seteaza_moderate" value="Da!">
  </form>
  <?
}
?>
</body></html>
```

Și, în final, fișierul care preia datele din formulare_moderare_opinii.php și efectuează modificările în baza de date:

prelucrare_moderare_comentarii.php

```
<?
include("autorizare.php");
include("admin_top.php");
/* modificare comentariu*/
if(isset($_POST['modifica']))
{
```

/* Verificăm să fie completate câmpurile. */

```
  if($_POST['nume_utilizator'] == "")
  {
    print "Nu ai completat numele
      utilizatorului!";
  }
  else if ($_POST['adresa_email'] == "")
  {
    print "Nu ai completat adresa de email!";
  }
  else if ($_POST['comentariu'] == "")
  {
    print "Câmpul Comentariu este gol!";
  }
  else
  {
```

/* câmpurile sunt completate, să facem modificarea în baza de date*/

```
  $sql = "UPDATE comentarii SET
    nume_utilizator='".$_POST['nume_utilizator']."',
    adresa_email='".$_POST['adresa_email']."',
    comentariu='".$_POST['comentariu']."'
    WHERE id_comentariu="
    .$_POST['id_comentariu'];
  mysql_query($sql);
  print "Comentariul a fost modificat!";
  }
}
```

/* ștergere comentariu */

```
if(isset($_POST['sterge']))
{
  $sql="DELETE FROM comentarii WHERE
    id_comentariu=".$_POST['id_comentariu'];
  mysql_query($sql);
  print "Comentariul a fost șters!";
}
```

/* setarea ultimului comentariu moderat în tabelul admin */

```
if(isset($_POST['seteaza_moderate']))
{
  $sql = "UPDATE admin SET
    ultimul_comentariu_moderat="
    .$_POST['ultimul_id'];
  mysql_query($sql);
  print "Valoarea a fost setată!";
}
?>
</body>
</html>
```

Testați funcționarea acestor scripturi, modificând și ștergând comentarii. De acum, la fiecare nouă vizită în secțiunea de administrare nu veți vedea decât comentariile noi, nemoderate. Dacă mai adăugați administratori ai site-ului, toți vor vedea cele mai noi comentarii, indiferent de cine le-a moderat pe ultimele.

Comenzi

Am văzut cum atunci când se face o nouă comandă putem primi un email de notificare. Ce ne facem însă dacă serverul nu poate trimite email? Sau dacă un virus șterge datele de pe partiția unde păstrăm email-urile? Putem face o pagină în secțiunea de administrare cu ajutorul căreia să monitorizăm comenzile, la fel cum moderăm comentariile. Avem o singură problemă: comenzile se fac în zile diferite, banii nu vin neapărat în ordinea în care avem comenzile în baza de date. Așadar nu putem pune pur și simplu un nou câmp în tabelul admin cu "ultima comandă onorată" deoarece pot exista comenzi mai vechi decât aceea, comenzi pentru care încă nu au venit banii. Va trebui atunci să facem o delimitare per comandă, indiferent de data sau id-ul acesteia. Vom face următorul lucru: în tabelul tranzactii (tabelul de comenzi) vom mai adăuga un câmp: comanda_onorata care poate avea două valori: 1 dacă tranzacția a fost efectuată cu succes (am primit banii, am trimis cărțile) sau 0 dacă comanda nu a fost (încă) onorată. Să adăugăm noul câmp în tabelul tranzactii, cu valoarea default 0:

```
ALTER TABLE tranzactii ADD
  comanda_onorata TINYINT UNSIGNED
  DEFAULT "0" NOT NULL;
SELECT * FROM tranzactii;
```

Să scriem și codul pentru fișierul comenzi.php. În acesta vom avea toate comenzile neonorate cu toate detaliile necesare: nume și adresă cumpărător, data, cărțile comandate, numărul și valoarea lor. De asemenea, pentru fiecare comandă vom avea două butoane: „Comandă onorată" pe care vom apăsa atunci când vom fi primit banii și trimis cărțile pentru comanda respectivă și „Anulează comanda" pe care vom apăsa atunci când comanda este foarte veche și utilizatorul nu a trimis încă banii sau atunci când utilizatorul ne anunță că nu mai dorește cărțile comandate). Iată codul:

comenzi.php

```
<?
include("autorizare.php");
include("admin_top.php");
?>
<h1>Comenzi</h1>
```

```
+---------------+-----------------+-----------------+-------------------+-----------------+
|id_tranzactie| data_tranzactie | nume_cumparator | adresa_cumparator | comanda_onorata |
+---------------+-----------------+-----------------+-------------------+-----------------+
|27           |20030304         | Ionescu Maria   | Minei 13, Petrosani |  0            |(neonorată)
|28           |20030304         | Georgescu Gheorghe | Medici nr. 17, Arad |  1         |(onorată)
|29           |20030304         | Romeo Ciupercescu | Zorilor 30, Campina |  0          |(neonorată)
+---------------+-----------------+-----------------+-------------------+-----------------+
```

```
<b>Comenzi încă neonorate</b>
<?
```

/* Afişăm lista comenzilor neonorate (WHERE comanda_onorata=0) din tabelul tranzactii: */

```
$sqlTranzactii = "SELECT id_tranzactie,
 DATE_FORMAT(data_tranzactie,'%d-%m-%Y')
 as data_tranzactie,
 nume_cumparator, adresa_cumparator
 FROM tranzactii WHERE
 comanda_onorata=0";
```

/* DATE_FORMAT este o funcţie a MySQL cu care putem formata o dată stocată într-un timestamp după cum dorim (în cazul de faţă, zz-ll-aaaa). Funcţia nu modifică nimic în baza de date ci doar afişează un TIMESTAMP într-un format mai uşor digerabil (04-03-2003 în loc de 20030304). */

```
$resursaTranzactii =
 mysql_query($sqlTranzactii);
while($rowTranzactie =
 mysql_fetch_array($resursaTranzactii))
{
?>
<form
  action="prelucrare_comenzi.php"
  method="POST">
Data comenzii:
<b><?=$rowTranzactie['data_tranzactie']?>
</b>
<div style="width:500px; border:1px
solid #ffffff; background-
color:#F9F1E7; padding:5px">
<b><?=$rowTranzactie['nume_cumparator']?>
</b><br>
<?=$rowTranzactie['adresa_cumparator']?>
  <TABLE border="1" cellpadding="4"
  cellspacing="0">
  <tr>
  <td align="center"><b>Carte</b></td>
  <td align="center"><b>Nr. buc</b></td>
  <td align="center"><b>Preţ</b></td>
  <td align="center"><b>Total</b></td>
  </tr>
  <?
```

/* şi, pentru fiecare tranzactie, afişăm cărţile comandate (titlul şi autorul), numărul şi valoarea lor: */

```
  $sqlCarti = "select titlu,
    nume_autor, pret, nr_buc from
    vanzari, carti, autori where
    carti.id_carte=vanzari.id_carte
    and carti.id_autor=autori.id_autor
    and id_tranzactie=".
    $rowTranzactie['id_tranzactie'];
  $resursaCarti=mysql_query($sqlCarti);
```

```
while($rowCarte =
  mysql_fetch_array($resursaCarti))
{
  print '<tr><td>'
    .$rowCarte['titlu'].' de
    '.$rowCarte['nume_autor'].'</td>
    <td align="right">
    '.$rowCarte['nr_buc'].'</td>
    <td align="right">
    '.$rowCarte['pret'].' </td>';
```

/* Calculăm totalul pentru această carte (pret * nr_buc)*/

```
    $total=$rowCarte['pret'] *
     $rowCarte['nr_buc'];
```

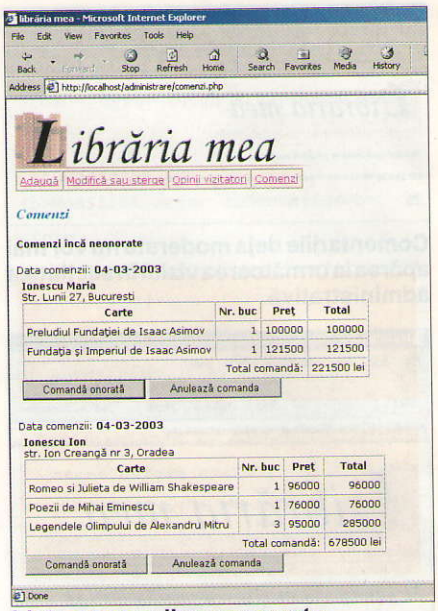
/* Afişăm acest total şi apoi îl adunăm la totalul general pentru această comandă: */

```
    print '<td
    align="right">'.$total.'</td>
    </tr>';
    $totalGeneral = $totalGeneral +
     $total;
  }
  ?>
  <tr>
  <TD colspan="3"
   align="right">Total comandă:</td>
  <td><?=$totalGeneral?> lei</td>
  </tr>
  </table>
<INPUT type="hidden"
  name="id_tranzactie" value=
  "<?=$rowTranzactie['id_tranzactie']?>">
<INPUT type="submit"
  name="comanda_onorata"
  value="Comandă onorată">
<INPUT type="submit"
  name="anuleaza_comanda"
  value="Anulează comanda">
</div>
</form>
<?
}
?>
</body>
</html>
```

Nu vom mai trece prin formulare de confirmare (ştiţi deja cum să faceţi acest lucru), facem direct un fişier care operează modificările în baza de date:

prelucrare_comenzi.php

```
<?
include("autorizare.php");
include("admin_top.php");
print "<h1>Comenzi</h1>";
```



Lista comenzilor neonorate.

/* comanda a fost onorată*/

```
if(isset($_POST['comanda_onorata']))
{
```

/* setăm câmpul comanda_onorata=1 în tabelul tranzactii pentru această comandă */

```
  $sql = "UPDATE tranzactii SET
    comanda_onorata=1 WHERE
    id_tranzactie=".$_POST['id_tranzactie'];
  mysql_query($sql);
```

/* şi afişăm un mesaj*/

```
  print "Comanda a fost onorată!";
}
```

/* comanda a fost anulată */

```
if(isset($_POST['anuleaza_comanda']))
{
```

/* ştergem tranzacţia (din tabelul tranzactii) şi cărţile comandate (din tabelul vanzari)*/

```
  $sqlTranzactii = "DELETE FROM
    tranzactii WHERE id_tranzactie="
    .$_POST['id_tranzactie'];
  mysql_query($sqlTranzactii);
  $sqlCarti = "DELETE FROM vanzari
    WHERE id_tranzactie="
    .$_POST['id_tranzactie'];
  mysql_query($sqlCarti);
```

/* şi afişăm un mesaj*/

```
  print "Comanda a fost anulată!";
}
?>
</body>
</html>
```

Aceasta a fost secţiunea de administrare. Acum aveţi un site funcţional, sigur, uşor de navigat şi mai ales uşor de administrat.

Tips and tricks

Întrebări frecvente

În acest capitol veți afla cum să creați aplicații on-line sigure și cum să vă protejați de neplăcerile găurilor de securitate.

Am adunat în acest capitol răspunsurile la cele mai arzătoare întrebări puse de programatorii începători. Elemente care la prima vedere ar putea părea inofensive pot fi periculoase dacă sunt privite din alt unghi. Variabilele globale, lipsa ghilimelelor magice și extensiile fișierelor pot fi folosite împotriva dumneavoastră.

Măsuri de securitate

Securizarea aplicațiilor on-line este un subiect vast abordat în nenumărate cărți. Aici vom vedea doar câteva din metodele pe care un programator PHP le are la îndemână pentru a scrie aplicații mai sigure.

Regula numărul unu a securității on-line este: nu va încredeți niciodată în utilizator. Întotdeauna verificați datele trimise către server și „curățați-le" înainte de a le utiliza. Pentru aceasta trebuie să considerați câteva posibile „găuri":

1. Variabilele globale

Având variabilele globale ON puteți accesa datele trimise prin formulare mai simplu: $variabila în loc de $_GET['variabila'], la fel și pentru POST sau FILES. Dacă scriptul nu este foarte bine gândit, variabilele globale pot prezenta un risc major de securitate. Din acest motiv php.ini este distribuit cu globals=off în ultimele versiuni.

2. Ghilimele „magice"

Dacă în php.ini magic_quotes_gpc sunt OFF folosiți funcția addslashes pentru a preceda ghilimelele din datele trimise de utilizatori cu caracterul \. Dacă magic_quotes_gpc sunt ON, PHP adaugă automat caracterul \ înainte de ghilimelele dar dacă ele sunt OFF, ghilimelele din input vă pot crea probleme serioase. Ca exemplu, să presupunem că interogarea SQL de verificare a numelui și parolei pentru înregistrarea pe site este SELECT * FROM users WHERE nume='$nume' and parola='$parola' și dacă interogarea este executată cu succes, utilizatorul este logat. În acest caz, folosind parola ' OR '1 = 1 oricine poate avea acces pe site deoarece interogarea SELECT * FROM users WHERE nume='un nume oarecare' AND PAROLA='' OR '1 = 1' este executată cu succes și returnează un rezultat (toate înregistrările din baza de date, de fapt).

Mai mult, atunci când vă așteptați ca baza de date să returneze un singur rând, verificați acest lucru și nu dacă interogarea s-a executat cu succes:

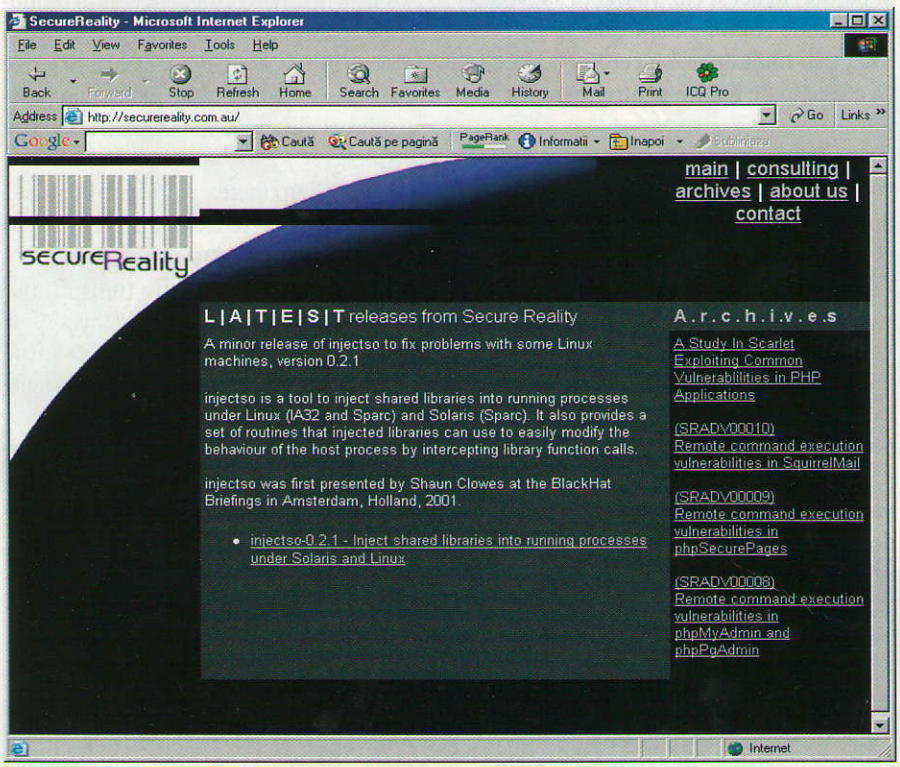
```
/* așa nu */
$sql = "SELECT * FROM users WHERE
nume='$nume' and parola='$parola'";
$resursa = mysql_query($sql);
if($resursa)
{
... autentificat...
}
/* așa da */
$sql = "SELECT * FROM users WHERE
nume='$nume' and parola='$parola'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) == 1)
{
... autentificat...
}
```

Notă: magic_quotes_gpc sunt implicit ON în php.ini pentru a vă proteja de astfel de atacuri, dar este bine să verificați înainte de a renunța la addslashes.

3. Includere

Încercați să evitați includerea „vizibilă" a fișierelor în forma http://site.ro/fisier.php?file=cutare.html pentru a include fișiere în cadrul unei pagini. Cu puțină neatenție din partea voastră atacatorul ar putea accesa astfel informații sensibile din cadrul sistemului.

Nu includeți fișiere străine. PHP poate „include" fișiere aflate pe alte servere decât cel care rulează dacă setarea **URL fopen wrappers** este activată în php.ini. În exemplul de mai sus, un atacator ar fi putut accesa adresa http://site.ro/



Pe site-ul Secure Reality găsiți mai multe articole privind securitatea PHP.

fisier.php?file=http://www.raul.ro/scriptrau.php pentru include în fişier un script localizat pe alt server şi astfel obţine acces către toate resursele sistemului la care are acces PHP, putând executa comenzi de sistem, afişa informaţii confidenţiale sau şterge baza de date. Nu permiteţi includerea fişierelor din altă parte decât de pe serverul vostru. Setaţi allow_url_fopen=Off în php.ini.

4. Formulare

Folosiţi metoda POST în formulare atunci când informaţia din acestea urmează să fie introdusă în baza de date. Am să vă dau un exemplu: să presupunem că avem un formular cu ajutorul căruia utilizatorii pot introduce părerile lor despre site, formular care este prelucrat de către un fisier userinput.php:

formular.html

```
<form action="userinput.php"
method="GET">
<textarea name="parerea_mea">
</textarea>
<input type="submit">
</form>
```

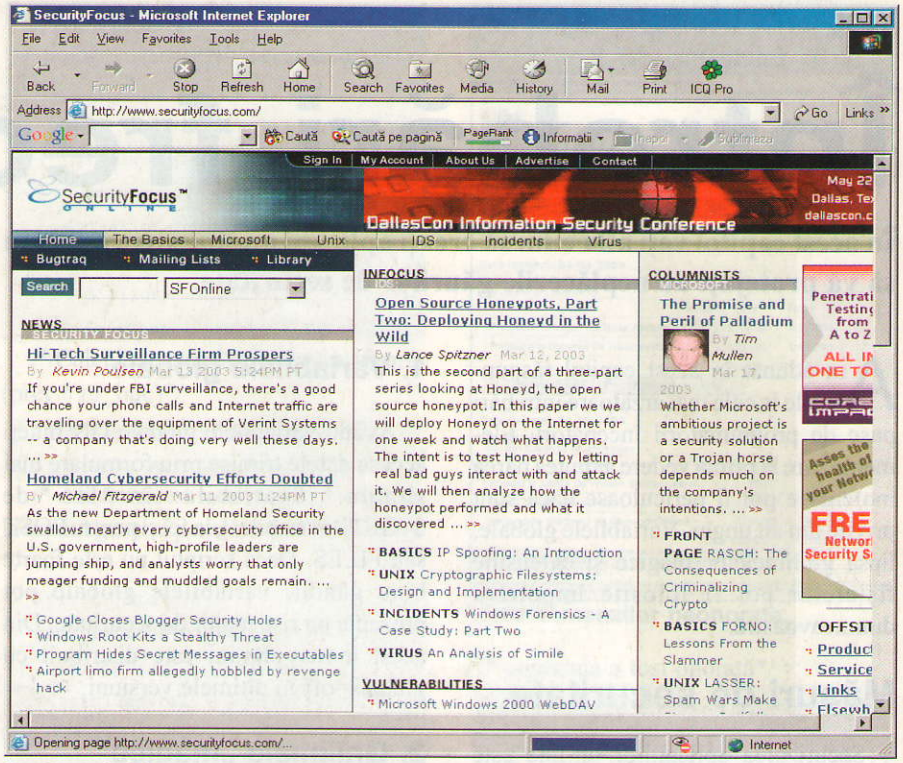
userinput.php

```
<?
mysql_connect(...);
mysql_query("insert into pareri
values('$parerea_mea')");
?>
```

Dacă variabilele globale sunt OFF în php.ini sau metoda de transmitere a formularului este GET, un utilizator rău intenţionat ar putea accesa adresa http://site.ro/userinput.php?parereamea=blabla şi ar introduce comentariul „blabla" în baza voastră de date fără să treacă propriu-zis prin site.

Dacă vă gândiţi cumva că asta n-ar fi chiar o problema atât de mare, imaginaţi-vă că „răufacatorul" scrie un script care accesează adresa fără oprire timp de câteva ore. Dacă legătura lui la Internet e bună, e posibil ca a doua zi dimineaţă să vă treziţi fără spaţiu pe server, cu câteva milioane de comentarii care spun acelasi lucru: „blabla".

Verificarea provenientei cererilor către server este foarte importantă şi în alt caz: formularele de login. Cineva care vă ştie numele de utilizator ar putea să încerce (şi, cu ceva noroc, chiar să reuşească) să vă găsească parola foarte uşor



Pe SecurityFocus.com sunt prezentate cele mai noi găuri de securitate întâlnite în aplicaţii.

în acest mod. Având la dispoziţie un dicţionar, intuiţia sau un generator de parole se poate accesa foarte usor şi rapid adresa www.site.ro/login.php?username=NumeleTau&parola=aaa, www.site.ro/login.php?username=tu&parola=aab, www.site.ro/login.php?username=tu&parola=aac şi aşa mai departe până la descoperirea combinaţiei valide de nume şi parolă. În acest caz va trebui să puneţi o protecţie suplimentară care să nu permită mai mult de trei încercări consecutive eşuate de logare pentru un nume de utilizator.

Această problemă se rezolvă rapid folosind sesiunile. În momentul în care un utilizator trimite numele şi parola către login.php, putem seta o variabilă de sesiune $_SESSION['login_count'] care să ţină minte numărul încercărilor. Când valoarea acesteia trece de 3 (trei încercări nereuşite) nici măcar nu mai interogaţi baza de date pentru a verifica a patra încercare.

Variabila $_SESSION['login_count'] va rămane în memorie cât browserul este deschis şi sesiunea activă (opţiunea implicită a PHP de menţinere a sesiunilor active este de o oră). Dacă atacatorul aşteaptă o oră sau îşi inchide browserul, sesiunea va fi închisă şi va putea încerca de alte trei ori să se logheze.

De cele mai multe ori această masură de siguranţă este suficientă pentru a preveni încercarile de aflare a parolelor utilizatorilor. Închiderea şi deschiderea browserului pentru a încerca de 3 ori este suficient de descurajantă ca să determine un hacker să îşi caute de lucru în altă parte.

5. Extensii

O practică obişnuită este de a acorda extensia .inc fişierelor care conţin biblioteci de funcţii ce urmează a fi incluse şi folosite în cod. PHP nu parsează fişierele cu extensia .inc şi dacă acestea sunt apelate direct ele sunt trimise plain text către browser.

Evitaţi expunerea informaţiilor dacă daţi acestor fişiere extensia .php - astfel ele vor fi rulate în loc să fie afişate. Nu puneţi informaţii sensibile (precum numele şi parola cu care vă conectaţi la serverul MySQL) în fişiere cu extensia .inc, .txt sau .html care pot fi accesate şi văzute.

Folosiţi pentru acestea fişiere cu extensia .php care, dacă sunt accesate direct, vor fi rulate fără să afişeze informaţiile conţinute în ele.

Cum spuneam, acestea sunt doar câteva din metodele pe care le puteţi folosi pentru a face aplicaţii sigure. Siguranţă absolută nu există însă prevenirea poate fi salvatoare. Mai multe informaţii privind securitatea puteţi găsi la www.securereality.com.au şi www.securityfocus.com.